Unión Internacional de Telecomunicaciones

UIT-T Z.145

(03/2006)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE Z: LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

Técnicas de descripción formal – Notación de prueba y de control de prueba

Notación de pruebas y de control de pruebas versión 3: Interfaz de control

Recomendación UIT-T Z.145

RECOMENDACIONES UIT-T DE LA SERIE Z

LENGUAJES Y ASPECTOS GENERALES DE SOPORTE LÓGICO PARA SISTEMAS DE TELECOMUNICACIÓN

TÉCNICAS DE DESCRIPCIÓN FORMAL	
Lenguaje de especificación y descripción	Z.100–Z.109
Aplicación de técnicas de descripción formal	Z.110–Z.119
Gráficos de secuencias de mensajes	Z.120–Z.129
Lenguaje ampliado de definición de objetos	Z.130–Z.139
Notación de prueba y de control de prueba	**Z.140–Z.149**
Notación de requisitos de usuarios	Z.150–Z.159
LENGUAJES DE PROGRAMACIÓN	
CHILL: el lenguaje de alto nivel del UIT-T	Z.200–Z.209
LENGUAJE HOMBRE-MÁQUINA	
Principios generales	Z.300–Z.309
Sintaxis básica y procedimientos de diálogo	Z.310–Z.319
LHM ampliado para terminales con pantalla de visualización	Z.320–Z.329
Especificación de la interfaz hombre-máquina	Z.330–Z.349
Interfaces hombre-máquina orientadas a datos	Z.350–Z.359
Interfaces hombre-máquina para la gestión de las redes de telecomunicaciones	Z.360–Z.379
CALIDAD	
Calidad de soportes lógicos de telecomunicaciones	Z.400–Z.409
Aspectos de la calidad de las Recomendaciones relativas a los protocolos	Z.450–Z.459
MÉTODOS	
Métodos para validación y pruebas	Z.500–Z.519
SOPORTE INTERMEDIO	
Entorno del procesamiento distribuido	Z.600–Z.609

Para más información, véase la Lista de Recomendaciones del UIT-T.

Recomendación UIT-T Z.145

Notación de pruebas y de control de pruebas versión 3: Interfaz de control

Resumen

En esta Recomendación se especifican las interfaces de control para las implementaciones del sistema de notación de pruebas y de control de pruebas (TTCN-3). Gracias a la interfaz de control TTCN-3 (TCI) se obtiene la adaptación recomendada para la gestión, el procesamiento de componentes de prueba y la codificación/decodificación de un sistema de prueba en una plataforma de prueba determinada. En esta Recomendación se definen las interfaces mediante un conjunto de operaciones que son independientes del lenguaje que se utilice.

Las interfaces se definen de tal manera que sean compatibles con la Rec. UIT-T Z.140. En las definiciones de interfaz de esta Recomendación se emplea el lenguaje de definición de interfaz (IDL) CORBA para especificar completamente la TCI. En las cláusulas 8 y 9 se indican las correspondencias de lenguaje entre la especificación abstracta y los lenguajes de utilización previstos: Java y ANSI-C. En el anexo A se incluye un resumen de la especificación de interfaz basada en el IDL.

Orígenes

La Recomendación UIT-T Z.145 fue aprobada el 16 de marzo de 2006 por la Comisión de Estudio 17 (2005-2008) del UIT-T por el procedimiento de la Recomendación UIT-T A.8.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Asamblea Mundial de Normalización de las Telecomunicaciones (AMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la AMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

La observancia de esta Recomendación es voluntaria. Ahora bien, la Recomendación puede contener ciertas disposiciones obligatorias (para asegurar, por ejemplo, la aplicabilidad o la interoperabilidad), por lo que la observancia se consigue con el cumplimiento exacto y puntual de todas las disposiciones obligatorias. La obligatoriedad de un elemento preceptivo o requisito se expresa mediante las frases "tener que, haber de, hay que + infinitivo" o el verbo principal en tiempo futuro simple de mandato, en modo afirmativo o negativo. El hecho de que se utilice esta formulación no entraña que la observancia se imponga a ninguna de las partes.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB en la dirección http://www.itu.int/ITU-T/ipr/.

ÍNDICE

Página

Notación de pruebas y de control de pruebas versión 3: Interfaz de control

1 Alcance

En esta Recomendación se especifican las interfaces de control para las implementaciones del sistema de notación de pruebas y de control de pruebas, versión 3 (TTCN-3). La interfaz de control TTCN-3 (TCI) permite obtener una adaptación normalizada para la gestión, el procesamiento de componentes de prueba y la codificación/decodificación de un sistema de prueba en una plataforma de prueba determinada. En esta Recomendación se definen las interfaces mediante un conjunto de operaciones que son independientes del lenguaje que se utilice.

Las interfaces se definen para que sean compatibles con la norma TTCN-3 (véanse las referencias correspondientes). En las definiciones de interfaz se emplea el lenguaje de definición de interfaz (IDL, *interface definition language*) CORBA para especificar completamente la TCI. En las cláusulas 8 y 9 se indican las correspondencias de lenguaje entre esta especificación abstracta y los lenguajes de utilización previstos: Java y ANSI-C. En el anexo A se incluye un resumen de la especificación de interfaz basada en el IDL.

2 Referencias

Las siguientes Recomendaciones del UIT-T y otras referencias contienen disposiciones que, mediante su referencia en este texto, constituyen disposiciones de la presente Recomendación. Al efectuar esta publicación, estaban en vigor las ediciones indicadas. Todas las Recomendaciones y otras referencias son objeto de revisiones por lo que se preconiza que los usuarios de esta Recomendación investiguen la posibilidad de aplicar las ediciones más recientes de las Recomendaciones y otras referencias citadas a continuación. Se publica periódicamente una lista de las Recomendaciones UIT-T actualmente vigentes. En esta Recomendación, la referencia a un documento, en tanto que autónomo, no le otorga el rango de una Recomendación.

[1] Recomendación UIT-T Z.144 (2006), *Notación de pruebas y de control de pruebas versión 3: Interfaz de ejecución*.

[2] Recomendación UIT-T Z.140 (2006), *Notación de pruebas y de control de pruebas versión 3: Lenguaje núcleo*.

[3] Recomendación UIT-T Z.143 (2006), *Notación de pruebas y de control de pruebas versión 3: Semántica operacional*.

[4] Recomendación UIT-T X.290 (1995), *Metodología y marco de las pruebas de conformidad de interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Conceptos generales*.

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

[5] Recomendación del W3C (2004), *XML Schema Part 0: Primer*.
NOTA – Véase http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[6] Recomendación del W3C (2004), *XML Schema Part 1: Structures*.
NOTA – Véase http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[7] Recomendación del W3C (2004), *XML Schema Part 2: Datatypes*.
NOTA – Véase http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

3 Definiciones y abreviaturas

3.1 Definiciones

A los efectos de esta Recomendación, además de los siguientes términos y definiciones se aplican los dados en la Rec. UIT-T X.290 [4].

3.1.1 sucesión de pruebas abstractas (ATS, *abstract test suite*): Sucesión de pruebas compuesta por casos de pruebas abstractas.

3.1.2 códec: Entidad codificadora o decodificadora que sirve para, respectivamente, codificar o decodificar la información que se ha de transmitir.

3.1.3 codificación/decodificación (CD): Entidad que administra el procesamiento del valor y del tipo, incluidas la codificación y la decodificación en el sistema de pruebas TTCN-3.

3.1.4 procesador de componentes (CH, *component handling*): Entidad que administra el procesamiento de componente de prueba en el sistema de pruebas TTCN-3.

3.1.5 puerto de comunicación: Mecanismo que facilita la comunicación entre componentes de prueba.

NOTA – Un puerto de comunicación se modela como una cola FIFO en el sentido de recepción. Los puertos pueden basarse en mensajes, en procedimientos o en una mezcla de ambos.

3.1.6 componente de control: Componente que hace funcionar la parte de control de un módulo TTCN-3.

3.1.7 sucesión de pruebas ejecutables (ETS, *executable test suite*): Véase la Rec. UIT-T X.290 [4].

3.1.8 información suplementaria de implementación de pruebas (*IXIT, implementation extra information for testing*): Véase la Rec. UIT-T X.290 [4].

3.1.9 adaptador de plataforma (PA, *platform adaptor*): Entidad que adapta el ejecutable TTCN-3 a una determinada plataforma de ejecución.

NOTA – El adaptador de plataforma crea, para un sistema de pruebas TTCN-3, una noción común de tiempo e implementa temporizadores, tanto explícitos como implícitos, así como también funciones externas.

3.1.10 interfaz de sistema de pruebas real: Véase la Rec. UIT-T X.290 [4].

3.1.11 sistema sometido a prueba (*SUT, system under test*): Véase la Rec. UIT-T X.290 [4].

3.1.12 adaptador SUT (SA): Entidad que adapta el funcionamiento de las comunicaciones TTCN-3 con el SUT, basándose en una interfaz de sistema de pruebas abstracto. Se encarga de implementar la interfaz de sistema real de pruebas.

3.1.13 notación de pruebas y control de pruebas (TTCN-3, *testing and test control notation*): Véase la Rec. UIT-T X.290 [4].

3.1.14 caso de prueba: Véase la Rec. UIT-T X.290 [4].

3.1.15 evento de prueba: Envío o recepción de información de prueba (mensaje o llamada de procedimiento) en un puerto de comunicación que es parte de la interfaz del sistema de pruebas. Puede tratarse también de eventos de expiración de temporizadores.

3.1.16 gestión de pruebas (TM, *test management*): Entidad que proporciona una interfaz de usuario, y que se encarga de la administración del sistema de pruebas TTCN-3.

3.1.17 registro de pruebas (TL, *test logging*): Entidad que proporciona información de registro acerca de la ejecución de pruebas (también incluye la información suministrada por el enunciado log TTCN-3).

3.1.18 gestión y control de pruebas (TMC, *test management and control*): Conjunto de entidades que permiten la gestión y el control de pruebas; está compuesto por la gestión de pruebas (TM, *test management*), el procesamiento de componentes (CH, *component handling*), el registro de pruebas (TL, test logging) y la Codificación/Decodificación (CD).

NOTA – El TMC es una implementación de la TCI.

3.1.19 sistema de pruebas: Véase la Rec. UIT-T X.290 [4].

3.1.20 interfaz de sistema de pruebas (TSI, *test system interface*): Componente de pruebas que proporciona la correspondencia entre los puertos disponibles en el sistema de pruebas (abstractas) TTCN-3 y las ofrecidas por el sistema de pruebas reales.

3.1.21 ejecutable TTCN-3 (TE, *TTCN-3 executable*): Parte de un sistema de pruebas que se encarga de la interpretación o la ejecución de una ETS TTCN-3.

3.1.22 interfaces de control TTCN-3 (TCI, *TTCN-3 control interfaces*): Tres interfaces que definen la interacción, en un sistema de pruebas, del ejecutable TTCN-3 con la gestión de pruebas, la codificación y la decodificación, y el procesamiento de componentes de prueba.

3.1.23 interfaz de ejecución TTCN-3 (TRI, *TTCN-3 runtime interface*): Interfaz que define la interacción, en un sistema de pruebas, del ejecutable TTCN-3 y los adaptadores de plataforma.

3.2 Abreviaturas, siglas o acrónimos

A los efectos de esta Recomendación, se utilizan las siguientes abreviaturas, siglas o acrónimos.

ATS Sucesión de pruebas abstractas (*abstract test suite*)

CD Codificación/Decodificación

CH Procesamiento de componentes (*component handling*)

ETS Sucesión de pruebas ejecutables (*executable test suite*)

IDL Lenguaje de definición de interfaz (*interface definition language*)

IXIT Información suplementaria de implementación para pruebas (*implementation extra information for testing*)

MSC Gráficos de secuencias de mensajes (*message sequence chart*)

MTC Componente de prueba principal (*main test component*)

OMG Grupo de gestión de objeto (*object management group*)

PA Adaptador de plataforma (*platform adaptor*)

PTC Componente de prueba paralelo (*parallel test component*)

SA Adaptador SUT (*SUT adaptor*)

SUT Sistema sometido a prueba (*system under test*)

TC Control de prueba (*test control*)

TCI Interfaces de control TTCN-3 (*TTCN-3 control interfaces*)

TE Ejecutable TTCN-3 (*TTCN-3 executable*)

TL Registro de pruebas (*test logging*)

TM Gestión de pruebas (*test management*)

TMC Gestión y control de pruebas (*test management and control*)

TRI Interfaz de ejecución TTCN-3 (*TTCN-3 runtime interface*)

TSI Interfaz de sistema de pruebas (*test system interface*)

TTCN-3 Notación de pruebas y control de pruebas versión 3 (*testing and test control notation version 3*)

4 Introducción

La presente Recomendación está compuesta por dos partes; en la primera se describe la estructura de una implementación de sistema de pruebas TTCN-3, mientras que en la segunda se presenta la especificación de interfaces de control TTCN-3.

En la primera parte se descompone el sistema de pruebas TTCN-3 en cuatro entidades principales, a saber:

- la gestión y control de pruebas (TMC);
- el ejecutable TTCN-3 (TE);
- el adaptador SUT (SA); y
- el adaptador de plataforma (PA).

La TMC se compone a su vez de tres entidades: la gestión de pruebas (TM), el codificador/decodificador (CD) y el procesador de componentes de prueba (CH). Además, se define la interacción entre ellas, es decir las interfaces correspondientes.

En la segunda parte de esta Recomendación se especifican las interfaces de control TTCN-3 (TCI), las cuales se definen en términos de las operaciones implementadas como parte de una entidad y llamadas por otras entidades de sistemas de pruebas. La especificación de interfaz define, para cada operación, las estructuras de información asociadas, el efecto esperado en el sistema de pruebas y toda restricción que se aplique al empleo de la operación. Cabe observar que estas especificaciones de interfaz sólo definen interacciones entre el TE y la TM, el TE y la CD, y el TE y el CH. Las interacciones entre el TE y el SA, y el TE y el PA se describen en la especificación de interfaz de ejecución TTCN-3 (Rec. UIT-T Z.144 [1]).

5 Conformidad

Una TCI que sea conforme a los sistemas de prueba TTCN-3 deberá, como mínimo, cumplir la especificación de interfaz definida en la presente Recomendación. La semántica TTCN-3 utilizada en el sistema de pruebas debe ser conforme a la semántica operacional definida en la Rec. UIT-T Z.143 [3]. Asimismo, se debe soportar una correspondencia de lenguaje. Así, por ejemplo, si un fabricante soporta Java, las llamadas de operación y las implementaciones de las TCI, que son parte del ejecutable TTCN-3, deben ser conformes a la correspondencia entre el IDL y Java especificada en esta Recomendación. En lo que toca a la interfaz de registro, es posible emplear la correspondencia XML en lugar de las correspondencias Java o C.

6 Estructura general de un sistema de pruebas TTCN-3

Conceptualmente, se puede concebir un sistema de pruebas TTCN-3 como un conjunto de entidades que interactúan entre sí, y cada una de las cuales realiza una funcionalidad específica de sistema de pruebas, a saber:

- gestionar la ejecución de pruebas;
- interpretar o ejecutar código TTCN-3 compilado;
- lograr la comunicación adecuada con el SUT;
- administrar tipos, valores y componentes de pruebas;
- implementar funciones externas; y
- administrar el funcionamiento de los temporizadores.

6.1 Entidades en un sistema de pruebas TTCN-3

En la figura 1 se muestra la estructura de una implementación de un sistema de pruebas TTCN-3.

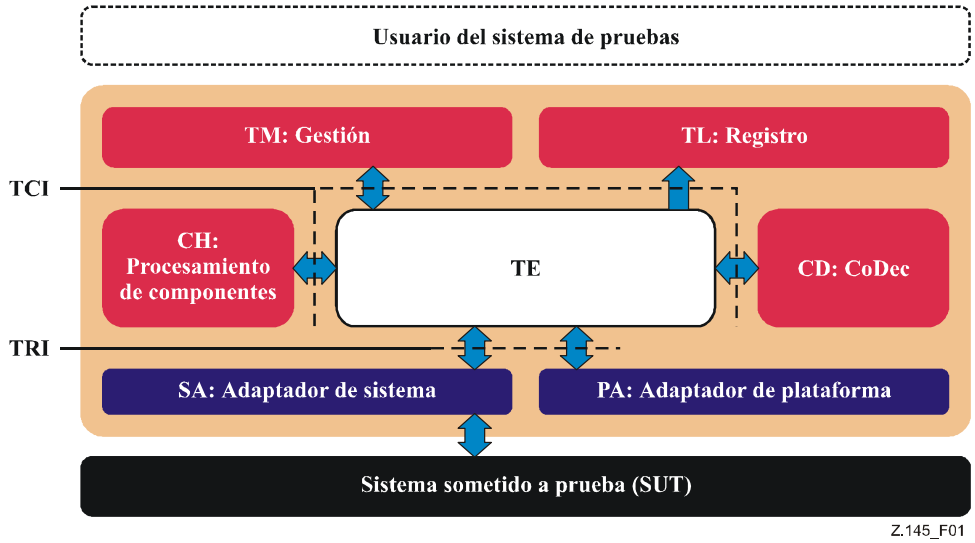

Figura 1/Z.145 – Estructura general de un sistema de pruebas TTCN-3

Como se indica en la figura 1, el ejecutable TTCN-3 (TE), también conocido como sucesión de pruebas ejecutables (ETS, *executable test suite*), interpreta y ejecuta módulos TTCN-3. Es posible identificar diversos elementos estructurales TE, a saber el de control, el de comportamiento, el de componentes, el de tipos, el de valores y el de colas. Los elementos estructurales de un TE representan las funcionalidades definidas en un módulo TTCN-3 o definidas por la propia norma TTCN-3 (Rec. UIT-T Z.140 [2]). Por ejemplo, el elemento estructural "Control" tiene que ver con la parte de control en un módulo TTCN-3, mientras que el elemento estructural "Colas" ("*Queues*") se refiere al requisito relativo al ejecutable TTCN-3 que consiste en que cada puerto de un componente de prueba ha de mantener su propia cola de puerto. Mientras que aquél se especifica dentro de un módulo TTCN-3, éste es un requisito de la especificación TTCN-3 propiamente dicha.

Con el fin de permitir una mejor definición de las interfaces de control TTCN-3, se proporciona una versión más refinada del TE (véase la figura 1). En general, en una implementación de un sistema de pruebas el TE será el código producido por un compilador TTCN-3 o por un interpretador de códigos TTCN-3.

Se puede ejecutar el TE de una manera centralizada o distribuida, es decir en un solo dispositivo de pruebas o en varios, respectivamente. Si bien en cada entidad estructural del TE se implementa un módulo TTCN-3 completo, tal vez sea necesario distribuir una misma entidad estructural entre varios dispositivos.

El TE implementa un módulo TTCN-3 en un nivel abstracto, y las otras entidades del sistema de pruebas TTCN-3 realizan dichos conceptos abstractos. Por ejemplo, no se puede implementar en un TE el concepto abstracto que consiste en enviar un mensaje o recibir la expiración de un temporizador. El resto del sistema de pruebas se encarga de la codificación del mensaje y de su envío a través de determinado medio físico, o de la medición del tiempo transcurrido y de establecer cuándo expira un temporizador, respectivamente.

En la Rec. UIT-T Z.144 [1] se definen los adaptadores SA y PA, así como su interacción con el TE. En la especificación de la TCI se define la interacción entre el TE y la TMC.

La interfaz de registro proporciona capacidades de registro a todos los elementos de la arquitectura del sistema de pruebas, es decir, gracias al registro TL, el TE, la TM, el CH, la CD, el SA y el PA pueden almacenar información acerca de la ejecución de prueba. En la figura 2 se muestra con más detalle el TL.

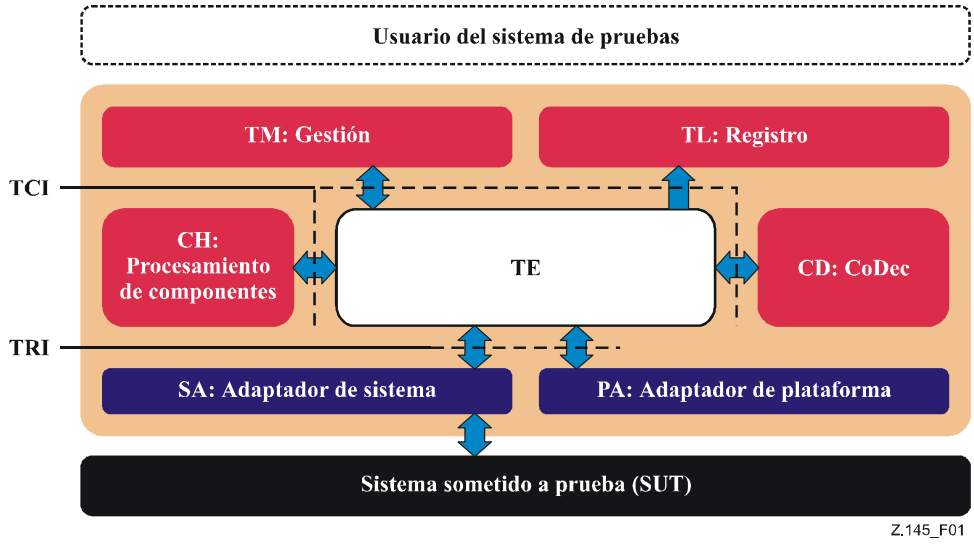

Figura 2/Z.145 – Descripción detallada del TL

6.1.1 Gestión y control de pruebas (TMC)

La entidad TMC incluye funcionalidades relacionadas con la gestión de:

- la ejecución de pruebas;
- los componentes; y
- la codificación y la decodificación.

6.1.1.1 Gestión de pruebas (TM)

La entidad TM se encarga de la gestión general de un sistema de pruebas. Tras la inicialización del sistema de pruebas, empieza la ejecución en la entidad TM. Esta entidad se encarga de la invocación adecuada de los módulos TTCN-3, es decir, del envío al TE de parámetros de módulo, como la información IXIT, si fuere necesario. En esta entidad también se suele implementar una interfaz de usuario del sistema.

6.1.1.2 Codificación y decodificación (CD)

La entidad CD se encarga de la codificación y decodificación de valores TTCN-3 en cadenas de bits adecuadas para ser enviadas a los sistemas sometidos a pruebas. El TE establece cuáles códecs se han de utilizar. Hace pasar la información TTCN-3 al codificador que venga al caso, para obtener la información codificada. La información recibida se decodifica entonces en la entidad CD empleando el decodificador correspondiente, que la traduce en valores TTCN-3.

6.1.1.3 Procesamiento de componentes (CH)

El TE puede estar distribuido entre varios dispositivos de pruebas. El CH realiza la comunicación entre las entidades distribuidas del sistema de pruebas. Gracias a la entidad CH es posible sincronizar las entidades del sistema de pruebas, que pueden estar distribuidas en varios nodos.

NOTA 1 – Se emplean como sinónimos las expresiones nodos y dispositivos de pruebas.

En la figura 3 se indica la estructura general de un sistema de pruebas distribuido entre varios nodos.

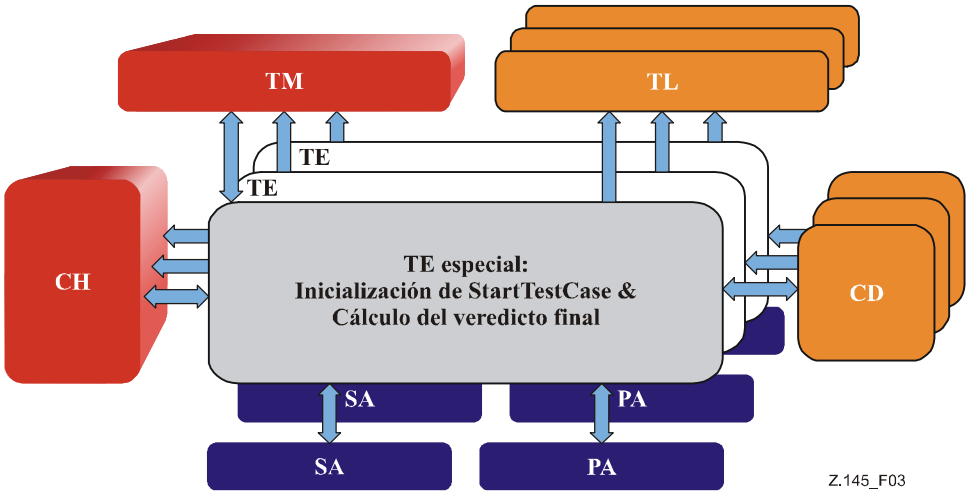

Figure 3/Z.145 – Estructura general de un sistema de pruebas TTCN-3 distribuido

Cada nodo de un sistema de pruebas tiene entidades TE, SA, PA, CD y TL. Las entidades CH y TM median la gestión de pruebas y el procesamiento de componentes de pruebas entre los TE en cada nodo. El TE que inicia un caso de prueba es un TE especial. Éste calculará el veredicto final del caso de prueba. Aparte de ello, todos los TE se procesan de la misma manera.

NOTA 2 – Un sistema de pruebas ejecutará a lo sumo un caso de pruebas al mismo tiempo. Es decir, un módulo TTCN-3 no puede ejecutar varios casos de pruebas simultáneamente.

El CH se encarga de controlar la creación de los componentes MTC, PTC y de control en los TE. Cabe observar la función particular que cumple el componente de sistema, que existe sólo conceptualmente y no como un componente de prueba que funcione realmente en un TE. Es posible distribuir entre varios nodos los puertos de sistema, es decir, los correspondientes al componente de sistema. Además, los componentes de prueba de diferentes nodos pueden tener acceso al mismo puerto físico del SUT, esto es, se pueden hacer corresponder al mismo puerto de la interfaz de sistema de pruebas.

EJEMPLO: Los TE pueden lograr el acceso a puertos reales distantes del SUT mediante apoderados locales.

La comunicación entre los componentes TTCN-3 se basa en mensajes o en procedimientos. El CH se encarga entonces de adaptar dicha comunicación entre los componentes TTCN-3 a la plataforma que emplee el sistema de pruebas. Esta entidad sabe cuáles son las conexiones entre los puertos de comunicación de los componentes de prueba TTCN-3, y hace pasar las operaciones de petición enviadas desde un componente TTCN-3 a otro. El componente que recibe puede estar en otro ejemplar del mismo TE ubicado en otro nodo, en cuyo caso notifica el TE acerca de todo evento de pruebas recibido, poniéndolo en una de las colas de puerto del TE.

El CH también participa en las comunicaciones basadas en procedimientos. En este caso, ha de distinguir entre los diferentes tipos de estas comunicaciones, es decir, llamada, respuesta y excepción, y deberá propagarlos correctamente hacia los TE donde se encuentran los componentes a los que van dirigidos. En el TE se procesa la semántica de las comunicaciones TTCN-3 basadas en procedimiento, es decir el efecto de la operación en cuestión sobre la ejecución del componente de pruebas TTCN-3.

Con el fin de poder distribuir los componentes de pruebas entre varios nodos, es necesaria una comunicación adicional. La comunicación relacionada con la gestión de componentes comprende varias partes, a saber la indicación de la creación de componentes de pruebas, del inicio de la ejecución de uno de ellos, de la distribución del veredicto, así como la indicación sobre la terminación del componente. La entidad CH no se encarga de implementar el comportamiento del componente TTCN-3, sino que implementa la comunicación entre varios componentes que a su vez han sido implementados por un TE.

6.1.1.4 Registro de pruebas (TL)

La entidad TL efectúa el registro de eventos de pruebas y se encarga de su presentación al usuario del sistema de pruebas. Almacena información acerca de la ejecución de la prueba, por ejemplo cuáles componentes de pruebas han sido creados, iniciados y terminados; cuál información se envía al SUT, se recibe de él y se hace corresponder con los modelos TTCN-3; cuáles temporizadores han sido iniciados, terminados o han expirado, etc.

6.1.2 Ejecutable TTCN-3 (TE)

La entidad TE ejecuta o interpreta un módulo TTCN-3. Conceptualmente, se puede descomponer en seis entidades que interactúan entre sí: las entidades de control, de comportamiento, de componente, de tipo, de valor y de cola, conforme a la Rec. UIT-T Z.144 [1]. En la presente Recomendación se emplea la terminología de dicha Recomendación.

6.1.3 Adaptador SUT (SA)

El SA es la implementación del adaptador del sistema sometido a pruebas (SA, *SUT adaptor*) definido en la Rec. UIT-T Z.144 [1]. En la presente Recomendación se emplea para el SA la terminología de dicha Recomendación.

6.1.4 Adaptador de plataforma (PA)

El PA es la implementación del adaptador de plataforma (PA, *platform adaptor*) definido en la Rec. UIT-T Z.144 [1]. En la presente Recomendación se emplea para el PA la terminología de dicha Recomendación.

6.2 Requisitos relativos a la ejecución en un sistema de pruebas TTCN-3

Cada llamada de operación TCI se tratará en la entidad llamante como una operación atómica. La entidad llamada, que implementa una operación TCI, tendrá que devolver el control a la llamante tan pronto haya logrado el objetivo previsto o cuando no se haya podido completar la operación con éxito. La entidad llamada no interrumpirá la puesta en marcha de una comunicación basada en procedimiento.

Como ya se dijo, no se supone que el sistema de pruebas TTCN-3 o las entidades propiamente dichas se hayan implementado en un solo ejecutable o proceso, ni que estén distribuidos entre diferentes procesos o, inclusive, entre diferentes dispositivos de prueba.

Toda implementación de una TCI habrá de satisfacer los requisitos anteriores.

7 Interfaz de control y operaciones TTCN-3

En esta cláusula se define un conjunto de tipos abstractos de información, que se emplean para representar comunicaciones entre el TE y la entidad TMC. Se definen también las operaciones TCI en términos de sus firmas, de cuándo se han de utilizar y de cuáles sean sus efectos sobre el sistema de pruebas TTCN-3.

Asimismo, esta definición incluye una descripción detallada de los parámetros iniciales requeridos en cada llamada de operación TCI y de sus correspondientes valores devueltos.

7.1 Resumen de la TCI

La TCI define, dentro de un sistema de pruebas TTCN-3, la interacción entre las siguientes entidades del TTCN-3: el TE, el CH, la TM, la CD y el TL. Esta interfaz permite al TE:

- gestionar la ejecución de pruebas;
- distribuir la ejecución de componentes de pruebas entre diversos dispositivos de prueba;
- codificar y decodificar información de prueba; y
- registrar información acerca de la ejecución de pruebas.

La TCI está compuesta por cuatro subinterfaces, a saber:

- **Interfaz de gestión de pruebas TCI (TCI-TM, *TCI test management interface*)**: Incluye todas las operaciones necesarias para gestionar la ejecución de pruebas, proporcionar parámetros de módulo y constantes externas y para suministrar un registro de eventos de prueba.
- **Interfaz de procesamiento de componentes TCI (TCI-CH, *TCI component handling interface*)**: Contiene las operaciones necesarias para la implementación, en un sistema de pruebas centralizado o distribuido, de la gestión de los componentes de prueba TTCN-3 y la comunicación entre ellos. Esto incluye operaciones para crear, iniciar e interrumpir componentes de prueba, establecer la conexión entre componentes TTCN-3, gestionar componentes de prueba y sus veredictos, y procesar comunicaciones basadas en mensajes y procedimientos entre dichos componentes.
- **Interfaz de codificación/decodificación TCI (TCI-CD, *TCI coding/decoding interface*)**: Incluye todas las operaciones necesarias para la obtención de códecs y el acceso a ellos, es decir los codificadores o los decodificadores, para la codificación de la información que se ha de enviar, que se define conforme al atributo de codificación TTCN-3, y para la decodificación de la información recibida.
- **Interfaz de registro de pruebas TCI (TCI-TL, *TLI test logging interface*)**: Incluye todas las operaciones necesarias para la obtención de información acerca de la ejecución de pruebas y para el control de su nivel de detalle.

Todas las interfaces son bidireccionales, con lo cual las partes llamante y llamada están en el TE y en la entidad TMC del sistema de pruebas. Las interfaces suministradas (aquellas operaciones ofrecidas por una interfaz al TE) y las operaciones requeridas (aquellas del TE que una interfaz requiere) se combinan respectivamente en la subinterfaz proporcionada y requerida para cada interfaz, es decir TCI-TM proporcionada/TCI-TM requerida, TCI-CH proporcionada/TCI-CH requerida, TCI-CD proporcionada/TCI-CD requerida y TCI-TL proporcionada/TCI-TL requerida.

7.1.1 Correlación entre invocaciones de operaciones TTCN-3 y TCI

Algunas invocaciones de operación TTCN-3 se correlacionan directamente con invocaciones de operación TCI, como se indica en el cuadro 1. Así, con el fin de poder efectuar la propagación de las operaciones TTCN-3 a través del sistema de prueba, ciertas operaciones TTCN-3 se correlacionan con un par petición de operación TCI y operación TCI. Para las demás invocaciones de operación TCI existe una correlación indirecta – son necesarias a la hora de implementar correctamente la semántica TTCN-3 de los conceptos subyacentes.

La correlación que se indica para las operaciones de comunicación TTCN-3 (es decir, `send`, `call`, `reply` y `raise`) sólo vale si estas operaciones se invocan en un puerto de componente de prueba conectado a otro puerto de componente de prueba. En la Rec. UIT-T Z.144 [1] se describe la correlación para el caso de operaciones de comunicación que se invocan en puertos de componente de prueba que se hacen corresponder con puertos de interfaz de sistema de pruebas.

Cuadro 1/Z.145 – Correlación entre invocaciones de operaciones TTCN-3 y TCI

Nombre de la operación TTCN-3	Nombre de la operación TCI	Nombre de la interfaz TCI
`send`	`tciSendConnected` (véase la nota 1)	TCI-CH proporcionada
	`tciSendConnectedBC` (véase la nota 2)	
	`tciSendConnectedMC` (véase la nota 3)	
	`tciEnqueueMsgConnected`	TCI-CH requerida
`call`	`tciCallConnected` (véase la nota 1)	TCI-CH proporcionada
	`tciCallConnectedBC` (véase la nota 2)	
	`tciCallConnectedMC` (véase la nota 3)	
	`tciEnqueueCallConnected`	TCI-CH requerida
`reply`	`tciReplyConnected` (véase la nota 1)	TCI-CH proporcionada
	`tciReplyConnectedBC` (véase la nota 2)	
	`tciReplyConnectedMC` (véase la nota 3)	
	`tciEnqueueReplyConnected`	TCI-CH requerida
`raise`	`tciRaiseConnected` (véase la nota 1)	TCI-CH proporcionada
	`tciRaiseConnectedBC` (véase la nota 2)	
	`tciRaiseConnectedMC` (véase la nota 3)	
	`tciEnqueueRaiseConnected`	TCI-CH requerida

Cuadro 1/Z.145 – Correlación entre invocaciones de operaciones TTCN-3 y TCI

Nombre de la operación TTCN-3	Nombre de la operación TCI	Nombre de la interfaz TCI
create	tciCreateTestComponentReq	TCI-CH proporcionada
	tciCreateTestComponent	TCI-CH requerida
start (a component)	tciStartTestComponentReq	TCI-CH proporcionada
	tciStartTestComponent	TCI-CH requerida
stop (a component)	tciStopTestComponentReq	TCI-CH proporcionada
	tciStopTestComponent	TCI-CH requerida
kill	tciKillTestComponentReq	TCI-CH proporcionada
	tciKillTestComponent	TCI-CH requerida
connect	tciConnectReq	TCI-CH proporcionada
	tciConnect	TCI-CH requerida
disconnect	tciDisconnectReq	TCI-CH proporcionada
	tciDisconnect	TCI-CH requerida
map	tciMapReq	TCI-CH proporcionada
	tciMap	TCI-CH requerida
unmap	tciUnmapReq	TCI-CH proporcionada
	tciUnmap	TCI-CH requerida
running	tciTestComponentRunningReq	TCI-CH proporcionada
	tciTestComponentRunning	TCI-CH requerida
alive	tciTestComponentAliveReq	TCI-CH proporcionada
	tciTestComponentAlive	TCI-CH requerida
done	tciTestComponentDoneReq	TCI-CH proporcionada
	tciTestComponentDone	TCI-CH requerida
killed	tciTestComponentKilledReq	TCI-CH proporcionada
	tciTestComponentKilled	TCI-CH requerida
mtc	tciGetMTCReq	TCI-CH proporcionada
	tciGetMTC	TCI-CH requerida
execute	tciTestCaseExecuteReq	TCI-CH proporcionada
	tciTestCaseExecute	TCI-CH requerida
log	tliLog	TCI-TL proporcionada
NOTA 1 – Para la comunicación unidifusión. NOTA 2 – Para la comunicación difusión. NOTA 3 – Para la comunicación multidifusión.		

7.2 Información TCI

En la especificación TCI se define un conjunto de tipos abstractos de información, los cuales describen, a un nivel muy general, qué tipo de información se ha de pasar entre una entidad llamante y una entidad llamada. Dichos tipos abstractos de información sirven para:

- establecer cómo se pasa información TTCN-3 de un TE a un codificador, codificar representaciones de valor TTCN-3 en cadenas de bits; y en el otro sentido,
- establecer cómo se ha de decodificar, de una cadena de bits a su representación de valor TTCN-3, la información pasada de un decodificador al TE.

Se define, para estos tipos abstractos de información, un conjunto de operaciones para que el codificador/decodificador pueda procesar la información.

En las correspondencias de lenguaje de las cláusulas 8 y 9 se definen, respectivamente, la representación concreta de estos tipos abstractos de información y los tipos de información básicos, como `string` y `boolean`.

Cabe observar que, para cualquier tipo de información de identificador, los valores han de ser únicos en determinada implementación del sistema de pruebas, entendiéndose por únicos el hecho de ser diferentes entre sí en todo momento.

Siendo así, se garantiza que a objetos diferentes, p. ej. dos temporizadores, no corresponda el mismo identificador, y que éstos no sean reutilizados.

7.2.1 Tipos abstractos de información general

Se definen los siguientes tipos abstractos de información y se los utiliza para la definición de operaciones TCI:

7.2.1.1 Gestión

`TciModuleIdType` Un valor de `TciModuleIdType` es el nombre de un módulo TTCN-3, conforme a lo especificado en la ATS TTCN-3. Este tipo abstracto se emplea para el procesamiento de módulo.

`TciModuleParameterIdType` Un valor de `TciModuleParameterIdType` es el nombre calificado de un parámetro de módulo TTCN-3, conforme a lo especificado en la ATS TTCN-3. Este tipo abstracto se emplea para el procesamiento de parámetro de módulo.

`TciTestCaseIdType` Un valor de `TciTestCaseIdType` es el nombre calificado de un caso de prueba TTCN-3, conforme a lo especificado en la ATS TTCN-3. Este tipo abstracto se emplea para el procesamiento de caso de prueba.

`TciModuleIdListType` Un valor del tipo `TciModuleIdListType` es una lista de `TciModuleIdType`. Este tipo abstracto se emplea al obtener una lista de módulos que son importados por un módulo TTCN-3.

`TciModuleParameterType` Un valor del tipo `TciModuleParameterType` es una estructura de `TciModuleParameterIdType` y `Value`. Este tipo abstracto se emplea para representar el nombre de parámetro y el valor por defecto de un parámetro de módulo.

`TciModuleParameterListType` Un valor del tipo `TciModuleParameterListType` es una lista de `TciModuleParameterType`. Este tipo abstracto se emplea al obtener los parámetros de módulo de un módulo TTCN-3.

`TciParameterType` Un valor del tipo `TciParameterType` incluye un `Value` TTCN-3 y un valor de `TciParameterPassingModeType` para representar el modo de paso de parámetros especificados en la ATS TTCN-3.

`TciParameterPassingModeType` Un valor del tipo `TciParameterPassingModeType` es `IN`, `INOUT` u `OUT`. Este tipo abstracto se emplea al iniciar un caso de prueba o cuando se indica el final de uno de ellos.

`TciParameterListType` Un valor del tipo `TciParameterListType` es una lista de `TciParameterType`. Este tipo abstracto se emplea al iniciar un caso de prueba o cuando se indica el final de uno de ellos.

`TciParameterTypeType` Un valor del tipo `TciParameterTypeType` es una estructura de tipo `Type` y `TciParameterPassingModeType`. Este tipo abstracto se emplea para representar el tipo el modo de paso de parámetros de un parámetro de caso de prueba.

`TciParameterTypeListType` Un valor del tipo `TciParameterTypeListType` es una lista de `TciParameterTypeType`. Este tipo abstracto se emplea para representar la lista de parámetros de un caso de prueba.

`TciTestComponentKindType` Un valor del tipo `TciTestComponentKindType` es un literal del conjunto de tipos de componentes de prueba TTCN-3, es decir `MTC`, `PTC`, `PTC_ALIVE` y `CONTROL`. Este tipo abstracto se emplea en el procesamiento de componentes.

`TciTypeClassType` Un valor del tipo `TciTypeClassType` es un literal del conjunto de clases de tipos TTCN-3 tales como booleano, punto flotante, registro, etc. Este tipo abstracto se emplea en el procesamiento de valores.

7.2.1.2 Comunicación

`TciBehaviourIdType` Un valor del tipo `TciBehaviourIdType` identifica una función de comportamiento TTCN-3.

Se toman de la Rec. UIT-T Z.144 [1] los siguientes tipos abstractos adicionales de información con el prefijo Tri: `TriPortIdType`, `TriPortIdListType`, `TriComponentIdType`, `TriComponentIdListType`, `TriAddressType`, `TriAddressListType` y `TriMessageType`.

7.2.2 Valores y tipos abstractos de información TTCN-3

En esta cláusula se define el conjunto de tipos abstractos de información que constituyen la representación del tipo y valor TTCN-3. Se define la funcionalidad de cada tipo de información mediante un conjunto de operaciones conexas. Las operaciones que actúan sobre determinado tipo abstracto de información, o que lo utilizan, producen como resultado bien sea un valor del mismo tipo o bien de un tipo básico, como el `boolean`.

Se definen todas las operaciones utilizando el lenguaje de descripción de interfaz (IDL). En las cláusulas 8 y 9 se presentan correspondencias concretas de lenguaje para las operaciones sobre los tipos abstractos de información. En algunos lenguajes, se representa la aplicación de una operación sobre un tipo de información haciendo pasar el valor concreto (ya sea por valor o por referencia, dependiendo de cuál correspondencia se trate) como parámetro de la operación. Es posible que en otros lenguajes se escoja otro método de referencia del valor concreto, por ejemplo al considerar el valor como un objeto con respecto al cual se invoca un método correspondiente a la operación. De aquí en adelante, se utiliza el valor `null` para indicar la incapacidad de efectuar determinada tarea o señalar la ausencia de un parámetro facultativo. Se puede considerar este valor como un valor reservado que sirve para indicar un valor especial. En las correspondencias de lenguaje se definirá una representación concreta de dicho valor `null`.

La representación del tipo y valor abstractos TTCN-3 consta de dos partes, a saber:

- un tipo abstracto de información, `Type`, que representa todos los tipos TTCN-3 en un módulo TTCN-3;
- diversos tipos abstractos de información que representan valores TTCN-3, es decir valores TTCN-3 de un tipo TTCN-3 determinado. Pueden ser valores de tipos TTCN-3 predefinidos o tipos TTCN-3 definidos por el usuario.

El sistema de pruebas utiliza el tipo abstracto de información `Type` y los diversos tipos de información de valores abstractos para acceder a la información TTCN-3, evaluarla y codificarla. Por consiguiente, estos tipos abstractos de información definen el nivel de abstracción entre el TE TTCN-3 y el resto del sistema de pruebas que emplea las interfaces TCI.

7.2.2.1 Tipos abstractos de información TTCN-3

Los tipos TTCN-3, ya sea predefinidos o definidos por el usuario, se representarán, con arreglo a la presente Recomendación, mediante el tipo abstracto de información `Type`.

Se define un conjunto de operaciones para el tipo abstracto de información `Type`, con el fin de:

- hacer referencia a los tipos de información TTCN-3 predefinidos y los definidos por el usuario; y
- crear y mantener valores TTCN-3.

Se definen, para dicho tipo abstracto de información `Type`, las siguientes operaciones:

`TciModuleIdType getDefiningModule()`	Devuelve el identificador de módulo del módulo en el cual se define el tipo. Devuelve el valor `null` si el tipo es básico TTCN-3, p. ej. booleano, entero, etcétera.
`TString getName()`	Devuelve el nombre del tipo, como se define en el módulo TTCN-3.
`TciTypeClassType getTypeClass()`	Devuelve la clase de tipo del tipo correspondiente. Un valor de `TciTypeClassType` puede ser una de las siguientes constantes: `ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OBJID`, `OCTETSTRING`, `RECORD`, `RECORD_OF`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`, `VERDICT`.
`Value newInstance()`	Devuelve un valor nuevo del tipo determinado. Este valor inicial del valor creado es indefinido.
`TString getTypeEncoding()`	Devuelve el atributo de codificación de tipo definido en el módulo TTCN-3.
`TString getTypeEncodingVariant()`	Devuelve el atributo variante de codificación definido en TTCN-3, si lo hubiere. Si no se define un atributo variante de codificación, se devuelve el valor `null`.
`TStringseq getTypeExtension()`	Devuelve el atributo de extensión de tipo definido en el módulo TTCN-3.

7.2.2.2 Valores abstractos TTCN-3

Según la presente Recomendación, los valores TTCN-3 se representan en las interfaces TCI mediante numerosos tipos abstractos de información.

En la figura 4 se describe la jerarquía de los tipos abstractos de información, para los valores TTCN-3 (que para resumir se llamarán valores abstractos).

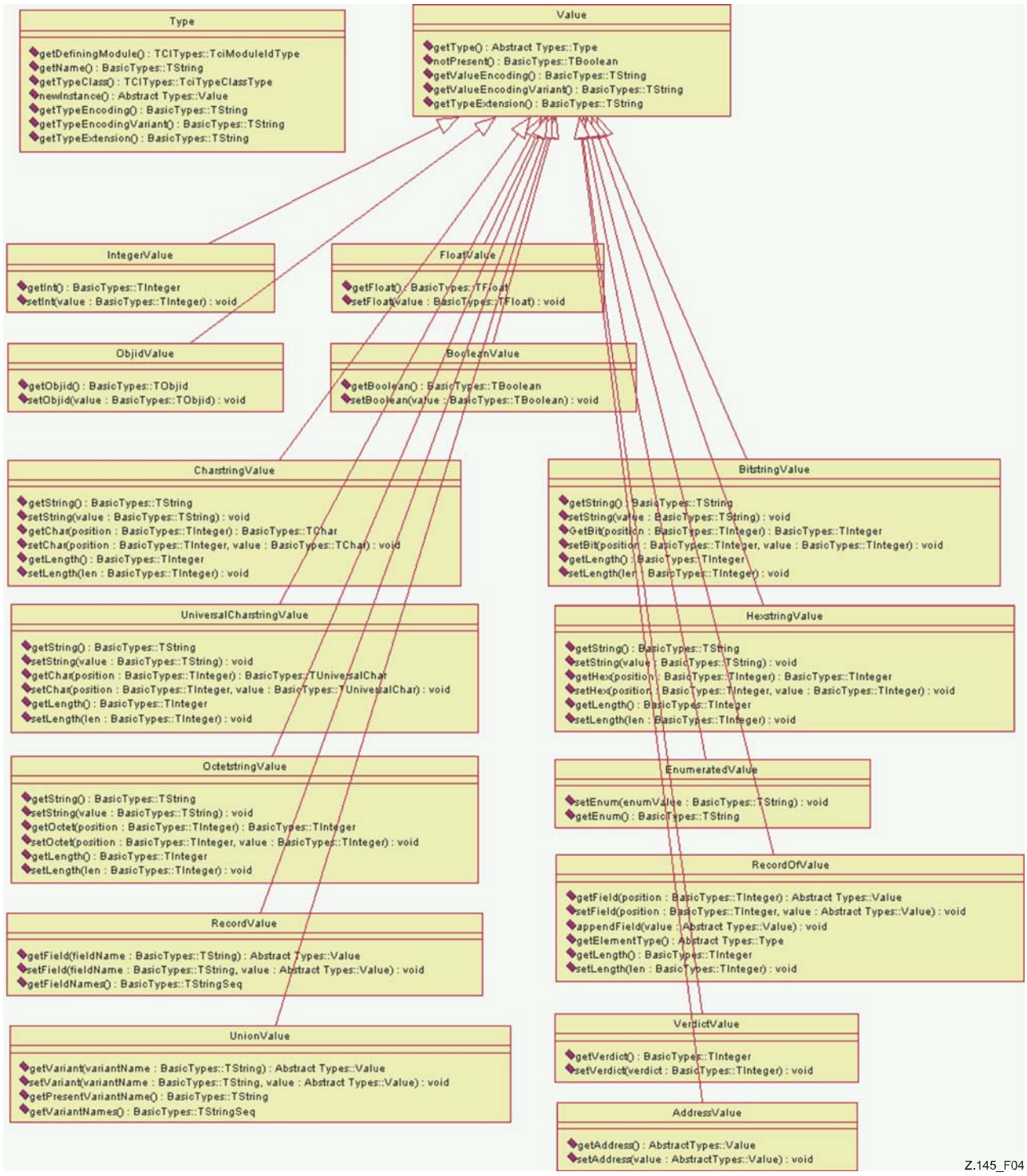

Figura 4/Z.145 – Jerarquía de valores abstractos

Como se muestra en la figura 4, todos los valore abstractos TTCN-3 comparten el mismo tipo abstracto de información de base, `Value`. Todas las operaciones que se definen en este tipo de información de base, se definen implícitamente para todos los tipos de valor abstracto que se derivan de él.

7.2.2.2.1 Tipo abstracto de información `Value`

Se definen las siguientes operaciones en el tipo abstracto de información de base, `Value`, mientras que las representaciones concretas se definen en las respectivas cláusulas dedicadas a las correspondencias de lenguaje:

`Type getType()`	Devuelve el tipo del valor especificado.
`TBoolean notPresent()`	Devuelve `true` si valor especificado es `omit`, o, de lo contrario, `false`.
`TString getValueEncoding()`	Devuelve el atributo de codificación de valor definido en TTCN-3, si lo hubiere. Si no se define atributo de codificación, devuelve el valor `null`.
`TString getValueEncodingVariant()`	Devuelve el atributo variante de codificación definido en TTCN-3, si lo hubiere. Si no se define un atributo variante de codificación, devuelve el valor `null`.

7.2.2.2.2 Tipo abstracto de información `IntegerValue`

El tipo abstracto de información `IntegerValue` se basa en el tipo abstracto de información `Value`, y representa valores TTCN-3 `integer`.

Se definen las siguientes operaciones en el tipo abstracto de información `IntegerValue`:

`TInteger getInt()`	Devuelve el valor entero de este entero TTCN-3.
`void setInt(in TInteger value)`	Fija este `IntegerValue` a `value`.

7.2.2.2.3 Tipo abstracto de información `FloatValue`

El tipo abstracto de información `FloatValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `float`.

Se definen las siguientes operaciones en el tipo abstracto de información `FloatValue`:

`TFloat getFloat()`	Devuelve el valor de punto flotante de este `float` TTCN-3.
`void setFloat(in TFloat value)`	Fija este `FloatValue` a `value`.

7.2.2.2.4 Tipo abstracto de información `BooleanValue`

El tipo abstracto de información `BooleanValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `boolean`.

Se definen las siguientes operaciones en el tipo abstracto de información `booleanValue`:

`TBoolean getBoolean ()`	Devuelve el valor booleano TTCN-3 `boolean`.
`void setBoolean(in TBoolean value)`	Fija este valor a `value`.

7.2.2.2.5 Tipo abstracto de información `ObjidValue`

El tipo abstracto de información `ObjidValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `objid`.

Se definen las siguientes operaciones en el tipo abstracto de información `ObjidValue`:

`TObjid getObjid()`	Devuelve el valor del id de objeto del TTCN-3 `objid`.
`void setObjid(in TObjid value)`	Fija este `ObjidValue` a `value`.

7.2.2.2.6 Tipo abstracto de información `CharstringValue`

El tipo abstracto de información `CharstringValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `charstring`. `TChar` es un carácter en un valor de cadena de caracteres.

Se definen las siguientes operaciones en el tipo abstracto de información `CharstringValue`:

`TString getString()`	Devuelve el valor de cadena del TTCN-3 `charstring`. La representación textual del TTCN-3 `charstring` vacío es `''`, y su longitud es cero.
`void setString(in TString value)`	Fija este `CharstringValue` a `value`.

`TChar getChar(in TInteger position)` Devuelve el valor carácter del TTCN-3 charstring en posición. Posición 0 indica el primer carácter del TTCN-3 `charstring`. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`void setChar(in TInteger position, in TChar value)`

Fija el carácter en position a un valor value. Los valores aceptados de position van desde `0` hasta `length - 1`.

`TInteger getLength()` Devuelve la longitud, expresada en caracteres, de CharstringValue, y cero si el valor de este CharstringValue es `omit`.

`void setLength(in TInteger len)` setLength reinicializa, en primer lugar, el valor CharstringValue a su valor inicial y luego pone la longitud de `CharstringValue` en caracteres igual a `len`.

7.2.2.2.7 Tipo abstracto de información `UniversalCharstringValue`

El tipo abstracto de información `UniversalCharstringValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `universal charstring`. `TUniversalChar` es un carácter en un valor de cadena de valores universal.

Se definen las siguientes operaciones en el tipo abstracto de información `UniversalCharstringValue`:

`TString getString()` Devuelve la representación textual de esta `UniversalCharstringValue`, como se define en TTCN-3.

`void setString(in TString value)` Fija el valor de esta `UniversalCharstringValue` con arreglo a la representación textual definida por `value`.

`TUniversalChar getChar(in TInteger position)`

Devuelve el valor de carácter universal del TTCN-3 `universal charstring` en posición. Posición 0 indica el primer `TUniversalChar` de la TTCN-3 `universal charstring`. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`void setChar(in TInteger position, in TUniversalChar value)`

Fija el carácter universal en posición a `value`. Los valores aceptados de posición van desde `0` hasta `length – 1`.

`TInteger getLength()` Devuelve la longitud de este valor de cadena de caracteres universal en caracteres universales, y es cero si el valor de esta charstring universal es `omit`.

`void setLength(in TInteger len)` setLength reinicializa, para comenzar, el UniversalCharstringValue poniéndolo en su valor inicial y luego fija la longitud de este `UniversalCharstringValue` en caracteres universales a `len`.

7.2.2.2.8 Tipo abstracto de información `BitstringValue`

El tipo abstracto de información `BitstringValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `bitstring`:

Se definen las siguientes operaciones en el tipo abstracto de información `BitstringValue`.

`TString getString()` Devuelve la representación textual de este `BitstringValue`, como se define en TTCN-3. Por ejemplo, la representación textual de `0101` es `"0101"B`. La representación textual de la TTCN-3 `bitstring` vacía es `""B`, y su longitud es cero.

`void setString(in TString value)` Fija el valor de este `BitstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de `BitstringValue` es `0101` si la representación textual en `value` es `"0101"B`.

`TChar getBit(in TInteger position)` Devuelve el valor `(0 | 1)` en la posición de esta cadena de bits TTCN-3 como un carácter. La posición 0 indica el primer bit de la cadena de bits TTCN-3. Los valores aceptados de `position` van desde `0` hasta `length - 1`.

`void setBit(in TInteger position, in TInteger value)`

Fija el bit en posición al valor `(0 | 1)`. La posición 0 indica el primer bit en este BitstringValue. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`TInteger getLength()` — Devuelve la longitud de este `BitstringValue` en bits, y es cero si el valor de esta `BitstringValue` es `omit`.

`void setLength(in TInteger len)` — setLength reinicializa, para comenzar, este BitstringValue poniéndolo en su valor inicial y luego fija la longitud de este `BitstringValue` en bits a `len`.

7.2.2.2.9 Tipo abstracto de información `OctetstringValue`

El tipo abstracto de información `OctetstringValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `octetstring`.

Se definen las siguientes operaciones en el tipo abstracto de información `OctetstringValue`:

`TString getString()` — Devuelve la representación textual de este `OctetstringValue`, como se define en TTCN-3. Por ejemplo, la representación textual de `0xCAFFEE` es `"CAFFEE"O`. La representación textual de la TTCN-3 `octetstring` vacía es `""O`, y su longitud es cero.

`void setString(in TString value)` — Fija el valor de este `OctetstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de `OctetstringValue` es `0xCAFFEE` si la representación textual en `value` es `"CAFFEE"O`.

`TChar getOctet(in TInteger position)` — Devuelve el valor `(0..255)` en la posición de esta cadena de octetos TTCN-3. La posición 0 indica el primer octeto de la cadena de octetos TTCN-3. Los valores aceptados de posición van desde `0` hasta `length – 1`.

`void setOctet(in TInteger position, in TInteger value)`

Fija el octeto en posición al valor `(0..255)`. La posición 0 indica el primer octeto de la cadena de octetos TTCN-3. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`TInteger getLength()` — Devuelve la longitud de este `OctetstringValue` en octetos, y es cero si el valor de este `OctetstringValue` es `omit`.

`void setLength(in TInteger len)` — setLength reinicializa, para comenzar, este OctetstringValue poniéndolo en su valor inicial y luego fija la longitud de este `OctetstringValue` en octetos a `len`.

7.2.2.2.10 Tipo abstracto de información `HexstringValue`

El tipo abstracto de información `HexstringValue` se basa en el tipo abstracto de información `Value`. Representa valores TTCN-3 `hexstring`.

Se definen las siguientes operaciones en el tipo abstracto de información `HexstringValue`:

`TString getString()` — Devuelve la representación textual de este `HexstringValue`, como se define en TTCN-3. Por ejemplo, la representación textual de `0xAFFEE` es `"AFFEE"H`. La representación textual de la TTCN-3 `hexstring` vacía es `""H`, y su longitud cero.

`void setString(in TString value)` — Fija el valor de este `HexstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de este `HexstringValue` es `0xAFFEE` si la representación en `value` es `"AFFEE"H`.

`TChar getHex(in TInteger position)` — Devuelve el valor `(0..15)` en la posición de esta cadena hexadecimal TTCN-3. La posición 0 indica la primera cifra hexadecimal de la cadena hexadecimal TTCN-3. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`void setHex(in TInteger position, in TInteger value)`

Fija la cifra hexadecinal en la posición al valor `(0..15)`. La posición 0 indica el primer octeto en la `hexstring`. Los valores aceptados de posición van desde `0` hasta `length - 1`.

`TInteger getLength()` — Devuelve la longitud de este `HexstringValue` en octetos, y es cero si el valor de este `HexstringValue` es `omit`.

`void setLength(in TInteger len)` — setLength reinicializa, para comenzar, este `HexstringValue` poniéndolo en su valor inicial y luego fija la longitud de este `HexstringValue` en cifras hex a `len`.

7.2.2.2.11 Tipo abstracto de información `RecordValue`

El tipo abstracto de información `RecordValue` se basa en el tipo abstracto de información `Value`. Especifica cómo obtener y fijar el tipo TTCN-3 `record`. El mismo tipo abstracto de información vale para valores cuya clase de tipo sea `SET`. Esta diferencia entre `record` y `set` sólo es importante durante la correspondencia.

Se definen las siguientes operaciones en el tipo abstracto de información `RecordValue`:

`Value getField(in TString fieldName)` — Devuelve el valor del campo denominado `fieldName`. El valor devuelto es el tipo base abstracto de información, `Value`, puesto que un campo de registro puede tener cualquier tipo definido en TTCN-3. Si no es posible obtener el campo a partir del registro, se devuelve el valor `null`.

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

Fija el campo `fieldName` del registro a `value`. No se ha de suponer nada acerca de cómo se almacena el campo en un registro. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido copiado. Se espera, entonces, que las modificaciones posteriores de `value` no serán consideradas en el registro.

`TStringSeq getFieldNames()` — Devuelve una secuencia de cadena de nombres de campo, la secuencia vacía si no hay ningún campo en el registro.

7.2.2.2.12 Tipo abstracto de información `RecordOfValue`

El tipo abstracto de información `RecordOfValue` se basa en el tipo abstracto de información `Value`. Especifica cómo obtener y fijar elementos en tipos TTCN-3 `record of`. El mismo tipo abstracto de información vale para valores cuya clase de tipo sea `SET_OF`. La diferencia entre `record of` y `set of` sólo es importante durante la correspondencia.

Se definen las siguientes operaciones en el tipo abstracto de información `RecordOfValue`:

`Value getField(in TInteger position)` — Devuelve el valor del `record of` en posición si la posición está entre `zero` y `length -1`, de lo contrario, devuelve `null`. El valor devuelto es el tipo base abstracto común, `Value`, puesto que un `record of` puede tener todo tipo de campos definidos en TTCN-3.

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

Fija el campo en la posición a `value`. Si posición es mayor que `(length - 1)` se amplía el record of hasta `(position + 1)`. Se ponen los elementos `record of` entre la posición original a `length` y `position - 1` a `omit`. No se supondrá nada acerca de cómo se almacenan los campos en un `record of`. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido asumido. Se espera, entonces, que las modificaciones posteriores de `value` no serán reflejadas en el `record of`.

`void appendField(in Value value)` — Añade el valor al final del `record of`, es decir en la posición `length`. No se supondrá nada acerca de cómo se almacenan los campos en un `record of`. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido asumido. Se espera, entonces, que las modificaciones posteriores de `value` no serán reflejadas en el `record of`.

`Type getElementType()` — Devuelve el `Type` de los elementos de este `record of`.

`TInteger getLength()`	Devuelve la longitud real del valor `record of`, y cero si el valor de `record of` es `omit`.
`void setLength(in TInteger len)`	Fija la longitud de `record of` a `len`. Si `len` es mayor que la longitud original, es porque hay elementos recientes que tienen el valor `omit`. Si `len` es menor o igual que la longitud original, se omite esta operación.

7.2.2.2.13 Tipo abstracto de información `UnionValue`

El tipo abstracto de información `UnionValue` se basa en el tipo abstracto de información `Value`. Especifica cómo obtener y fijar variantes en un tipo TTCN-3 `union`. Se representa el TTCN-3 `anytype` mediante un `UnionValue` donde la clase de tipo obtenida por la operación `getType()` es `ANYTYPE`. En la 7.2.2.2.1 se suministran detalles acerca de las clases de tipos.

Se definen las siguientes operaciones en el tipo abstracto de información `UnionValue`:

`Value getVariant(in TString variantName)`

Devuelve el valor de la unión TTCN-3 `variantName`, si `variantName` es igual al resultado de la operación `getPresentVariantName` o, de lo contrario, devuelve `null`. `variantName` indica el nombre de la variante unión definida en TTCN-3.

`void setVariant(in TString variantName, in Value value)`

Fija el `variantName` de la unión al valor correspondiente. Si no se ha definido `variantName` para esta unión, se omite esta operación. Si se escogió otra variante, se seleccionará la nueva en su lugar.

`TString getPresentVariantName()`	Devuelve una cadena que representa el nombre en vigor en la unión TTCN-3 del caso. Si no se escoge variante, se devuelve el valor `null`.
`TStringSeq getVariantNames()`	Devuelve una secuencia de cadena de nombres de variantes, y si la unión no tiene campos, devuelve el valor `null`. Si el `UnionValue` representa el TTCN-3 `anytype`, es decir, la clase del tipo obtenido mediante getType() es ANYTYPE, se devuelven los tipos TTCN-3 predefinidos o definidos por el usuario.

7.2.2.2.14 Tipo abstracto de información `EnumeratedValue`

El tipo abstracto de información `EnumeratedValue` se basa en el tipo abstracto de información `Value`. Especifica cómo obtener y fijar TTCN-3 `enumerated`.

Se definen las siguientes operaciones en el tipo abstracto de información `EnumeratedValue`:

`TString getEnum()`	Devuelve el identificador de cadena de este `EnumeratedValue`. Este identificador es igual al identificador que se indica en la especificación TTCN-3.
`void setEnum(in TString enumValue)`	Fija el enum a `enumValue`. Si `enumValue` no es un valor aceptado para esta enumeración, se omite la operación.

7.2.2.2.15 Tipo abstracto de información `VerdictValue`

El tipo abstracto de información `VerdictValue` se basa en el tipo abstracto de información `Value`. Especifica cómo obtener y fijar TTCN-3 `verdict`.

Se definen las siguientes operaciones en el tipo abstracto de información `VerdictValue`:

`TInteger getVerdict()`	Devuelve el valor entero de este `VerdictValue`. Donde el entero es una de las siguientes constantes: `ERROR`, `FAIL`, `INCONC`, `NONE`, `PASS`.
`void setVerdict(in TInteger verdict)`	Fija este `VerdictValue` a `verdict`. Obsérvese que siempre se puede fijar un `VerdictValue` a cualquiera de los veredictos mencionados. El `VerdictValue` no efectúa ningún cálculo de veredicto, como se define en TTCN-3. Por ejemplo, es válido fijar primero `VerdictValue` a `ERROR` y luego a `PASS`.

7.2.2.2.16 Tipo abstracto de información `AddressValue`

Se definen las siguientes operaciones en el tipo abstracto de información `AddressValue`. La representación concreta de dichas operaciones se define en las cláusulas de correspondencia de lenguaje:

`Value getAddress()`	Devuelve el valor de dirección, que ya no es la clase de tipo `ADDRESS` sino el tipo real empleado para la dirección.
`void setAddress(in Value value)`	Fija esta dirección a `value`.

7.2.3 Tipos abstractos de registro

7.2.3.1 Tipo abstracto de información `TciValueTemplate`

Se definen las siguientes operaciones en el tipo abstracto de información `TciValueTemplate`. La representación concreta de dichas operaciones se define en las secciones de correspondencia de lenguaje:

`boolean isOmit()`	Devuelve `true` si la plantilla es una plantilla omit.
`boolean isAny()`	Devuelve `true` si la plantilla es una any template.
`boolean isAnyOrOmit()`	Devuelve `true` si la plantilla es any template u omit.
`TString getTemplateDef()`	Devuelve la definición de dicha plantilla.

7.2.3.2 Tipo abstracto de información `TciNonValueTemplate`

Se definen las siguientes operaciones en el tipo abstracto de información `TciNonValueTemplate`. La representación concreta de dichas operaciones se define en las secciones de correspondencia de lenguaje:

`boolean isAny()`	Devuelve `true` si la plantilla es una any template.
`boolean isAll()`	Devuelve `true` si la plantilla es una all template.
`TString getTemplateDef()`	Devuelve la definición de dicha plantilla.

7.2.3.3 Lista de valores y tipos de correspondencia errónea

Se definen los siguientes tipos abstractos de información, que se utilizan para el registro de las diferencias entre valores y plantillas:

`TciValueList`	Un valor de `TciValueList` es una lista de valores.
`TciValueDifference`	Un valor de `TciValueDifference` es una estructura que contiene un valor, una plantilla, y una descripción del porqué de dicha diferencia.
`TciValueDifferenceList`	Un valor de `TciValueDifferenceList` es una secuencia de diferencias de valores.

7.3 Operaciones TCI

En esta cláusula se especifican las operaciones que ha de suministrar un ejecutable TTCN-3 (TE) a un sistema de pruebas (*operaciones requeridas*) y las funcionalidades que el sistema de pruebas tiene que proporcionar al ejecutable TTCN-3 (*operaciones proporcionadas*).

El empleo de las expresiones "requerida" y "proporcionada" refleja el hecho de que en la presente Recomendación se definen los requisitos para un ejecutable TTCN-3 desde el punto de vista del usuario. Este último "requiere" que el TE le permita cierta funcionalidad con el fin de crear un sistema completo de pruebas basado en TTCN-3, para lo cual el TE debe informar al usuario acerca de ciertos eventos para los que éste ha de "proporcionarle" esta posibilidad.

En todas las definiciones de operaciones de esta subcláusula se utiliza el lenguaje de definición de interfaz (IDL). En las cláusulas 8 y 9 se definen correspondencias concretas de lenguaje. En el anexo B se incluye otra correspondencia con XML para la interfaz de registro.

Para cada llamada de operación TCI, son obligatorios todos los parámetros *in*, *inout* y *out* enumerados en la definición de la operación en cuestión. El valor de un parámetro *in* lo especifica la entidad que llama. La entidad llamada se refiere a la dirección de la llamada. Cuando se trate de operaciones en una interfaz *requerida*, la entidad llamante es el sistema de pruebas mientras que la entidad llamada es el ejecutable TTCN-3. Si se trata de operaciones en una interfaz *proporcionada*, la entidad llamante es el ejecutable TTCN-3, y la llamada, el sistema de pruebas.

Igualmente, la entidad llamada especifica el valor de un parámetro *out*. De tratarse de un parámetro *inout*, la entidad llamante especifica primero un valor, que luego la entidad llamada puede reemplazar por uno nuevo. Cabe observar que si bien en TTCN-3 también se emplean los parámetros *in*, *inout* y *out* en las definiciones de firmas, las indicaciones utilizadas en la especificación IDL TCI no están relacionadas con aquellas en la especificación TTCN-3.

Las llamadas de operaciones deberían utilizar un valor reservado para indicar que no hay parámetros. Los valores reservados para estos tipos se definen en cada correspondencia de lenguaje y de aquí en adelante se denominarán valores `null`.

El valor `null` también se utilizará para indicar la incapacidad de efectuar determinada tarea.

Puerto que en esta cláusula sólo se especifican interfaces y no se proponen implementaciones concretas para ejecutar la funcionalidad especificada, se empleará el término entidad para identificar la parte de la implementación del sistema de pruebas que pone en marcha dicha interfaz y realiza la funcionalidad solicitada. Por ejemplo, en la operación tciSendConnected la entidad llamante es el TE, esto es, la parte de la implementación del sistema de pruebas que suministra la funcionalidad TE.

Se describen todas las funciones en la interfaz utilizando el siguiente modelo (plantilla). Se suprimen las descripciones que no valen para ciertas operaciones.

Firma	`Signature IDL`
Parámetros In (de entrada)	Descripción de la información que se hace pasar, como parámetros, a la operación desde la entidad llamante a la llamada.
Parámetros Out (de salida)	Descripción de la información que se hace pasar, como parámetros, a la operación desde la entidad llamada a la llamante.
Parámetros InOut	Descripción de la información que se hace pasar, como parámetros, a la operación desde la entidad llamante a la llamada y desde ésta como retorno a aquélla.
Valor devuelto	Descripción de la información que devuelve la operación a la entidad llamante.
Restricción	Descripción de las restricciones que puede haber cuando se invoca la operación.
Efecto	Comportamiento que ha de tener la entidad llamada antes de que pueda repetirse la operación.

7.3.1 La interfaz TCI-TM

La interfaz de gestión de pruebas TCI (TCI-TM, *test management interface*) describe las operaciones que un TE requiere implementar y las que ha de proporcionar al TE una implementación de gestión de pruebas (figura 5).

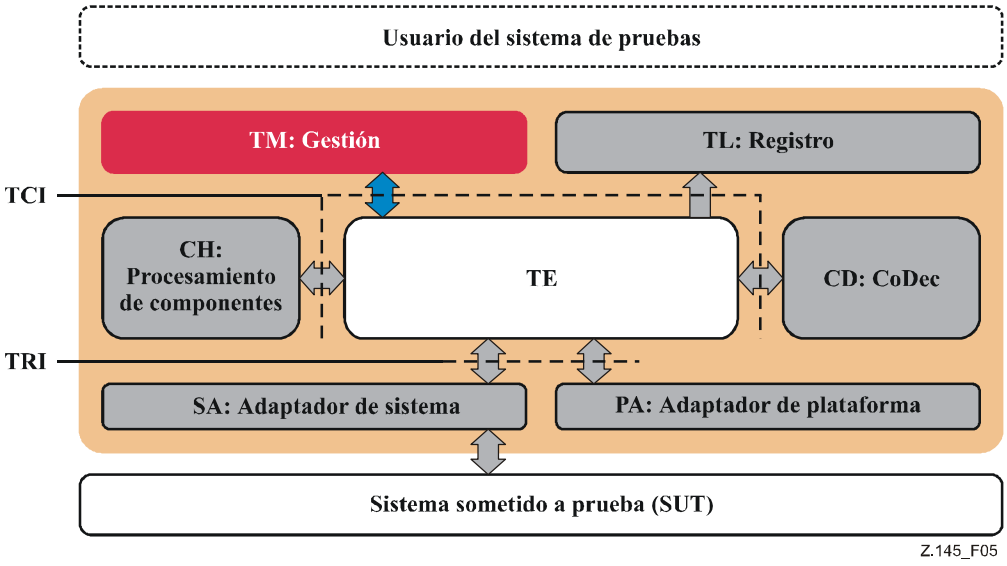

Figura 5/Z.145 – La interfaz TCI-TM

Una implementación de sistema de pruebas proporciona al usuario la gestión general de pruebas. Requiere del TE la presencia de operaciones para iniciar y detener ejecuciones de pruebas de un módulo TTCN-3 o de ciertos casos de pruebas de un módulo TTCN-3. Proporciona, a su vez, al TE las operaciones para resolver parámetros de módulo durante la ejecución y la indicación de que ésta ha terminado.

En la cláusula 10 se describe la utilización de las llamadas de operaciones y su ordenamiento secuencial bien sea por el TE o por la gestión de pruebas.

7.3.1.1 Requeridas por la TCI-TM

En esta cláusula se especifican las operaciones del TE que requiere el TM. Además de las que se especifican aquí, una gestión de pruebas ha de tener las operaciones requeridas en la interfaz TCI-CD.

7.3.1.1.1 tciRootModule

Firma	void `tciRootModule` (`in TciModuleIdType moduleName`)	
Parámetros In (de entrada)	`moduleName`	El `moduleName` indica los identificadores de módulo definidos en TTCN-3.
Valor devuelto	`void`	
Restricción	Se ha de emplear solamente si no se está ejecutando ni la parte de control ni un caso de pruebas.	
Efecto	`tciRootModule` selecciona el módulo indicado para la ejecución mediante una llamada posterior que utiliza `tciStartTestCase` o `tciStartControl`. El TE emitirá un `tciError` si no existe tal módulo.	

7.3.1.1.2 getImportedModules

Firma	`TciModuleIdListType getImportedModules()`
Valor devuelto	Lista de todos los módulos importados del módulo raíz. Se ordenan los módulos conforme a cómo figuran en el módulo TTCN-3. Si no hay módulos importados, se devuelve una lista vacía de módulos.
Restricción	Se ha de emplear solamente si se ha establecido antes un módulo raíz.
Efecto	El TE proporciona a la gestión una lista de módulos importados del módulo raíz. Si no hay módulos importados, se devuelve una lista vacía de módulos. Si el TE no puede proporcionar una lista, se devuelve el valor `null`.

7.3.1.1.3 tciGetModuleParameters

Firma	`TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName)`	
Parámetros In (de entrada)	`moduleName`	El `moduleName` indica los identificadores de módulo para los cuales se han de obtener los parámetros de módulo.
Valor devuelto	Lista de parámetros de módulo del módulo identificado. Se ordenan los parámetros conforme a cómo figuran en el módulo TTCN-3. Si no hay parámetros, se devuelve una lista vacía de parámetros de módulo.	
Restricción	Se ha de emplear solamente si se ha establecido antes un módulo raíz.	
Efecto	El TE proporciona a la gestión una lista de parámetros de módulo del módulo identificado. Si no hay parámetros de módulo, se devuelve una lista vacía de parámetros de módulo. Si el TE no puede proporcionar una lista, se devuelve el valor `null`.	

7.3.1.1.4 tciGetTestCases

Firma	`TciTestCaseIdListType tciGetTestCases ()`
Valor devuelto	Lista de todos los casos de prueba que son definidos o importados en el módulo raíz.
Restricción	Se ha de emplear solamente si se ha establecido antes un módulo raíz.
Efecto	El TE proporciona a la gestión una lista de casos de prueba. Si no hay casos de prueba, se devuelve una lista vacía de casos de prueba. Si el TE no puede proporcionar una lista, se devuelve el valor `null`.

7.3.1.1.5 tciGetTestCaseParameters

Firma	`TciParameterTypeListType tciGetTestCaseParameters (in TciTestCaseIdType testCaseId)`	
Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
Valor devuelto	Lista de todos los tipos de parámetro del caso de pruebas dado. Los tipos de parámetros se ordenan conforme a cómo figuran en la firma TCN-3 del caso de pruebas. Si no hay parámetros, se devuelve una lista vacía de parámetros.	
Restricción	Se ha de emplear solamente si se ha establecido antes un módulo raíz.	
Efecto	El TE proporciona a la gestión una lista de tipos de parámetro del caso de pruebas dado. Si no hay parámetros de caso de pruebas, se devuelve una lista vacía de parámetros de caso de pruebas. Si el TE no puede proporcionar una lista, se devuelve el valor `null`.	

7.3.1.1.6 tciGetTestCaseTSI

Firma	`TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)`	
Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
Valor devuelto	Lista de puertos de sistema del caso de pruebas dado que han sido declarados en la definición del componente de sistema para el caso de pruebas, es decir, los puertos TSI. Si no se ha definido explícitamente un componente de sistema para el caso de pruebas, la lista contendrá todos los puertos de comunicación del componente de pruebas MTC. Los puertos se ordenan conforme a su aparición en la respectiva declaración de tipo de componente TTCN-3. Si no hay puertos de sistema, se devuelve una lista vacía, es decir una de longitud cero.	
Restricción	Se ha de emplear solamente si se ha establecido antes un módulo raíz.	
Efecto	El TE proporciona a la gestión una lista de puertos de sistema del caso de pruebas dado. Si no hay puertos de sistema, se devuelve una lista de puertos vacía. Si el TE no puede proporcionar una lista, se devuelve el valor `null`.	

7.3.1.1.7 tciStartTestCase

Firma	`void tciStartTestCase(in TciTestCaseIdType testCaseId, in TciParameterListType parameterList)`	
Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
	`parameterList`	Lista de `Values` en la que cada uno de ellos define un parámetro de la lista de parámetros especificada en la definición del caso de pruebas TTCN-3. Los parámetros de la `parameterList` se ordenan conforme a cómo aparezcan en la firma TTCN-3 del caso de pruebas. Si no hay que pasar parámetros, se hace pasar bien sea el valor `null` o una `parameterList` vacía, es decir una lista de longitud cero.
Valor devuelto	`void`	
Restricción	Se ha de invocar solamente si se ha escogido antes un módulo. Sólo se podrán pasar `testCaseIds` cuando los casos de pruebas hayan sido declarados en los módulos TTCN-3 actualmente seleccionados. No se pueden iniciar los casos de pruebas que hayan sido importados en un módulo referenciado. A fin de poder iniciar casos de pruebas importados, se debe escoger antes el módulo referenciado utilizando la operación `tciRootModule`.	
Efecto	`tciStartTestCase` inicia un caso de pruebas en el módulo en vigor con los parámetros dados. El TE emitirá un `tciError` de no haber dichos casos de pruebas. Todos los parámetros de caso de pruebas *in* e *inout*, en una `parameterList`, contienen el `Value`. Todos los parámetros de caso de pruebas *out*, en una `parameterList` han de contener el valor `null`, ya que sólo son pertinentes cuando termina el caso de pruebas.	

7.3.1.1.8 tciStopTestCase

Firma	`void tciStopTestCase()`
Valor devuelto	`void`
Restricción	Se ha de invocar solamente si se ha escogido antes un módulo.
Efecto	tciStopTestCase para el caso de pruebas que se está ejecutando. Si el TE no está ejecutando un caso de pruebas, se omite la operación. Cuando se esté ejecutando la parte de control, `tciStopTestCase` interrumpirá el caso de prueba actualmente en ejecución, es decir el que se acaba de indicar mediante la operación *proporcionada* `tciTestCaseStarted`. De haber una parte de control de ejecución, ésta seguirá con la ejecución como si el caso de pruebas hubiera parado normalmente y devuelto un veredicto `ERROR`.

7.3.1.1.9 tciStartControl

Firma	`TriComponentId tciStartControl()`
Valor devuelto	Un `TriComponentId` que representa la componente de pruebas en la que se ejecuta la parte de control de módulo. Si el TE no puede iniciar la parte de control del módulo escogido, se devolverá el valor `null`.
Restricción	Se ha de invocar solamente si se ha escogido antes un módulo.
Efecto	Inicia la parte de control del módulo escogido. La parte de control iniciará casos de pruebas TTCN-3 como se describe en TTCN-3. Mientras ejecuta la parte de control, el TE invocará las operaciones *proporcionadas* `tciTestCaseStarted` y `tciTestCaseTerminated` para cada caso de pruebas que ha sido inicializado y que ha terminado. Tras la terminación de la parte de control, el TE invocará la operación *proporcionada* `tciControlPartTerminated`.

7.3.1.1.10 tciStopControl

Firma	`void tciStopControl()`
Valor devuelto	`void`
Restricción	Se ha de invocar solamente si se ha escogido antes un módulo.
Efecto	`tciStopControl` detiene la ejecución de la parte de control. Si no se está ejecutando ninguna parte de control, se omite la operación. Cuando se haya iniciado directamente un caso de pruebas, se detendrá la ejecución del caso de pruebas en vigor, como si se hubiera invocado la operación `tciStopTestCase`.

7.3.1.2 Proporcionada por la TCI-TM

En esta cláusula se especifican las operaciones que la TM ha de proporcionar al TE.

7.3.1.2.1 tciTestCaseStarted

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

Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
	`parameterList`	Lista de valores que forman parte de la firma del caso de pruebas. Los parámetros de la `parameterList` se ordenan conforme a su aparición en la declaración del caso de pruebas TTCN-3.
	`timer`	Valor float que representa la duración del temporizador de caso de pruebas.
Valor devuelto	`void`	
Restricción	Se ha de invocar solamente después de que se haya iniciado, utilizando las operaciones *requeridas* `tciStartControl` o `tciStartTestCase`, la parte de control del módulo o un caso de pruebas.	
Efecto	`tciTestCaseStarted` indica al TM que se ha iniciado un caso de pruebas con `testCaseId`. No se podrá distinguir si el caso ha sido iniciado explícitamente, utilizando la operación *requerida* `tciStartTestCase`,o implícitamente mientras se ejecutaba la parte de control.	

7.3.1.2.2 tciTestCaseTerminated

Firma	`void tciTestCaseTerminated(in VerdictValue verdict,` `in TciParameterListType parameterList)`	
Parámetros In (de entrada)	`verdict`	Veredicto final del caso de pruebas.
	`parameterList`	Lista de valores que forman parte de la firma del caso de pruebas. Los parámetros de la `parameterList` se ordenan conforme a su aparición en la declaración del caso de pruebas TTCN-3.
Valor devuelto	`void`	
Restricción	Se ha de invocar solamente después de que se haya iniciado, utilizando las operaciones *requeridas* `tciStartControl` o `tciStartTestCase`, la parte de control del módulo o un caso de pruebas.	
Efecto	El TE invocará esta operación para indicar a la TM que ha terminado el caso de pruebas que se está ejecutando en la MTC y que el veredicto final es `verdict`. Al invocarse una operación `tciTestCaseTerminated` todos los parámetros de caso de pruebas *out* e *inout* contienen `Values`. Todos los parámetros de caso de pruebas *in* contienen el valor `null`, puesto que sólo son importantes al inicio del caso de pruebas mas no en la respuesta a la llamada.	

7.3.1.2.3 tciControlTerminated

Firma	`void tciControlTerminated ()`
Valor devuelto	`void`
Restricción	Se ha de invocar solamente después de que se haya iniciado, utilizando la operación `tciStartControl`, la ejecución de módulo.
Efecto	El TE invocará esta operación para indicar a la TM que la parte de control del módulo escogido acaba de terminar la ejecución.

7.3.1.2.4 tciGetModulePar

Firma	`Value tciGetModulePar (in TciModuleParameterIdType parameterId)`	
Parámetros In (de entrada)	`parameterId`	Identificador del parámetro de módulo definido en TTCN-3.
Valor devuelto	Un valor.	
Restricción	Se invocará siempre que el TE necesite acceder al valor de un parámetro de módulo. Una sola vez se escogerá cada parámetro de módulo al que se accede: bien sea entre un par `tciStartTestCase` y `tciTestCaseTerminated`, cuando se haya iniciado explícitamente un caso de pruebas, o bien entre uno `tciStartControl` y `tciControlTerminated`, si se ha iniciado la parte de control de un módulo.	
Efecto	La TM proporciona al TE un `Value` para el `parameterId` indicado. Cada vez que se invoque `tciGetModulePar()` se devolverá el mismo valor a través de la ejecución de una parte de control. Si la gestión no puede proporcionar un valor TTCN-3, se devolverá el valor `null`.	

7.3.1.2.5 tciError

Firma	`void tciError(in TString message)`	
Parámetros In (de entrada)	`message`	Valor de cadena, es decir el error de mensaje
Valor devuelto	`void`	
Restricción	El TE puede invocarla en cualquier momento para indicar una situación de error que no se puede subsanar. Dicha situación puede ser indicada por el CH o por el CD, o puede ocurrir dentro del TE.	
Efecto	El TE indica que existe una situación de error que no se puede subsanar. `message` contiene una frase que explica el motivo, y que debe comunicarse al usuario del sistema de pruebas. Es potestad de la TM terminar la ejecución de casos de pruebas o partes de control, si las hubiere. La TM ha de tomar medidas explícitas para terminar inmediatamente la ejecución de pruebas.	

7.3.2 La interfaz TCI-CD

La interfaz de códec TCI (TCI-CD) describe las operaciones que se requiere que implemente un ejecutable TTCN-3 y las que debe proporcionar al TE una implementación de códec, para ciertos esquemas de codificación (figura 6).

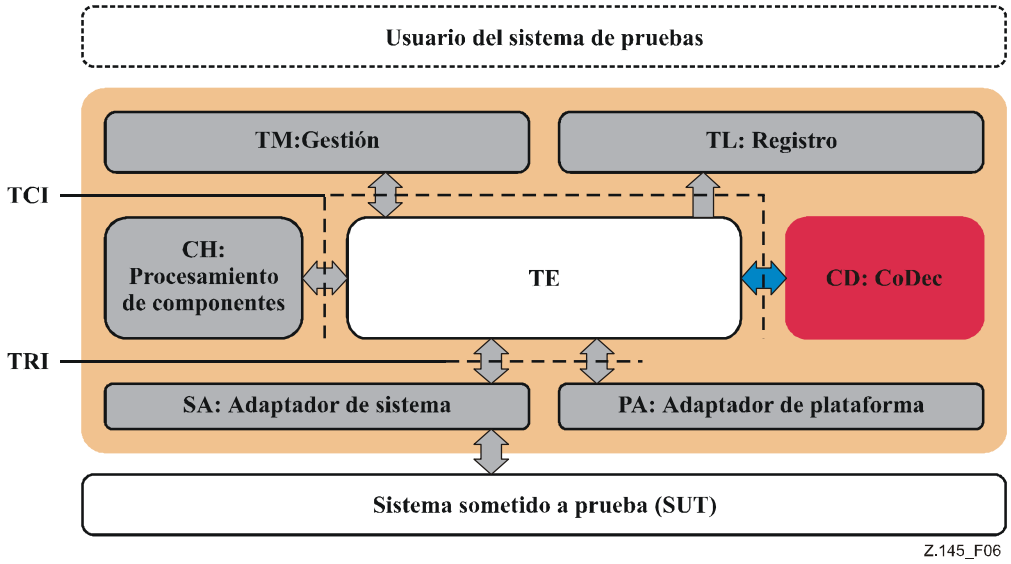

Figura 6/Z.145 – La interfaz TCI-CD

En una implementación de códec se codifican valores TTCN-3, conforme al atributo decodificación, transformándolos en una cadena de bits y se decodifican estas cadenas, con arreglo a las hipótesis de decodificación. Con el fin de poder decodificar una cadena de bits y obtener un valor TTCN-3, el CD requiere que el TE le facilite ciertas funcionalidades. A su vez, el CD proporciona al TE la funcionalidad de codificación y de decodificación.

En la cláusula 10 se explica la utilización de las invocaciones de operaciones, por parte del TE o del CD y su ordenamiento secuencial.

7.3.2.1 Requeridas por la TCI-CD

En esta cláusula se especifican las operaciones que el CD requiere del TE. Todas ellas se requieren también en las interfaces TCI-TM y TCI-CH.

7.3.2.1.1 getTypeForName

Firma	`Type getTypeForName(in TString typeName)`	
Parámetros In (de entrada)	`typeName`	Nombre TTCN-3 del tipo, como se define en el módulo TTCN-3. Los siguientes tipos son reservados, en cuyo caso se devolverá un tipo predefinido: "integer" "float" "bitstring" "hexstring" "octetstring" "charstring" "universal charstring" "boolean" "verdicttype" "objid" typeName ha de ser el nombre completo de tipo calificado, es decir module.typeName
Valor devuelto	Tipo que representa el tipo TTCN-3 solicitado.	
Restricción	---	
Efecto	Devuelve un tipo que representa un tipo TTCN-3. Se pueden obtener del TE tipos TTCN-3 predefinidos, utilizando palabras clave TTCN-3. En este caso, `typeName` indica tipos básicos TTCN-3 como "charstring", "bitstring", etc. Devuelve el valor `null` cuando no se pueda devolver el tipo solicitado. Obsérvese que no se pueden obtener `anytype` y `address` con módulos puestos a `null`. Si bien se trata de tipos predefinidos, pueden ser diferentes entre módulos. Por ejemplo, `address` puede ser el tipo predefinido sin modificar, o un tipo definido por un usuario en un módulo. No se pueden redefinir otros tipos predefinidos.	

7.3.2.1.2 getInteger

Firma	`Type getInteger()`
Valor devuelto	Ejemplar de `Type` que representa un tipo entero TTCN-3.
Efecto	Construye y devuelve un tipo entero básico TTCN-3.

7.3.2.1.3 getFloat

Firma	`Type getFloat()`
Valor devuelto	Ejemplar de `Type` que representa un tipo float TTCN-3.
Efecto	Construye y devuelve un tipo float básico TTCN-3.

7.3.2.1.4 getBoolean

Firma	`Type getBoolean()`
Valor devuelto	Ejemplar de `Type` que representa un tipo booleano TTCN-3.
Efecto	Construye y devuelve un tipo booleano básico TTCN-3.

7.3.2.1.5 getObjid

Firma	`Type getObjid()`
Valor devuelto	Ejemplar de `Type` que representa un tipo de id de objeto TTCN-3.
Efecto	Construye y devuelve un tipo de id de objeto básico TTCN-3.

7.3.2.1.6 getCharstring

Firma	`Type getCharstring ()`
Valor devuelto	Ejemplar de `Type` que representa un tipo charstring TTCN-3.
Efecto	Construye y devuelve un tipo charstring básico TTCN-3.

7.3.2.1.7 getUniversalCharstring

Firma	`Type getUniversalCharstring ()`
Valor devuelto	Ejemplar de `Type` que representa un tipo charstring universal TTCN-3.
Efecto	Construye y devuelve un tipo charstring universal TTCN-3.

7.3.2.1.8 getHexstring

Firma	`Type getHexstring ()`
Valor devuelto	Ejemplar de `Type` que representa un tipo hexstring TTCN-3.
Efecto	Construye y devuelve un tipo hexstring básico TTCN-3.

7.3.2.1.9 getBitstring

Firma	`Type getBitstring()`
Valor devuelto	Ejemplar de `Type` que representa un tipo bitstring TTCN-3.
Efecto	Construye y devuelve un tipo bitstring básico TTCN-3.

7.3.2.1.10 getOctetstring

Firma	`Type getOctetstring ()`
Valor devuelto	Ejemplar de `Type` que representa un tipo octetstring TTCN-3.
Efecto	Construye y devuelve un tipo octetstring básico TTCN-3.

7.3.2.1.11 getVerdict

Firma	`Type getVerdict()`
Valor devuelto	Ejemplar de `Type` que representa un tipo verdict TTCN-3.
Efecto	Construye y devuelve un tipo verdict básico TTCN-3.

7.3.2.1.12 tciErrorReq

Firma	`void tciErrorReq(in TString message)`	
Parámetros In (de entrada)	`Message`	Valor de cadena, es decir la frase de error en la que se describe el problema.
Valor devuelto	`void`	
Restricción	Se ha de invocar siempre que haya ocurrido una situación de error.	
Efecto	Se notificará al TE acerca de una situación de error insubsanable en el CD y se le reenviará la indicación de error.	

7.3.2.2 Proporcionadas por la TCI-CD

En esta cláusula se especifican las operaciones que la TM ha de proporcionar al TE.

7.3.2.2.1 decode

Firma	`Value decode(in TriMessageType message, in Type decodingHypothesis)`	
Parámetros In (de entrada)	`message`	El mensaje codificado que se va a decodificar.
	`decodingHypothesis`	Las hipótesis en las que se basará la decodificación.
Valor devuelto	Devuelve el valor decodificado, si el valor es de tipo compatible como el `decodingHypothesis`, o de lo contrario el valor `null`.	
Restricción	Se debe invocar siempre que el TE tenga que decodificar un valor codificado. El TE debe efectuar la decodificación inmediatamente después de haber recibido un valor codificado, o puede, por razones de rendimiento, posponerla hasta el acceso real del valor codificado.	
Efecto	En esta operación se decodifica el `message` conforme a las reglas de codificación y se devuelve un valor TTCN-3. Se empleará la `decodingHypothesis` para establecer si es posible decodificar el valor codificado. Cuando una regla de codificación no sea autosuficiente, es decir, cuando el mensaje codificado no contenga inherentemente su tipo, se utilizará la `decodingHypothesis`. Cuando se pueda efectuar la decodificación sin recurrir a la hipótesis, se devolverá el valor `null` si el tipo determinado a partir del mensaje es incompatible con la hipótesis de decodificación.	

7.3.2.2.2 encode

Firma	`TriMessageType encode(in Value value)`	
Parámetros In (de entrada)	`value`	Valor que se ha de codificar.
Valor devuelto	Devuelve un `TriMessage` codificado mediante la regla especificada de codificación.	
Restricción	Se debe invocar siempre que el TE tenga que codificar un valor.	
Efecto	Devuelve `TriMessage` codificado conforme a las reglas de codificación.	

7.3.3 La interfaz TCI-CH

La interfaz de procesamiento de componentes TCI (TCI-CH) describe las operaciones que debe implementar un ejecutable TTCN-3 y las que debe proporcionar al TE una implementación de procesamiento de componentes (figura 7).

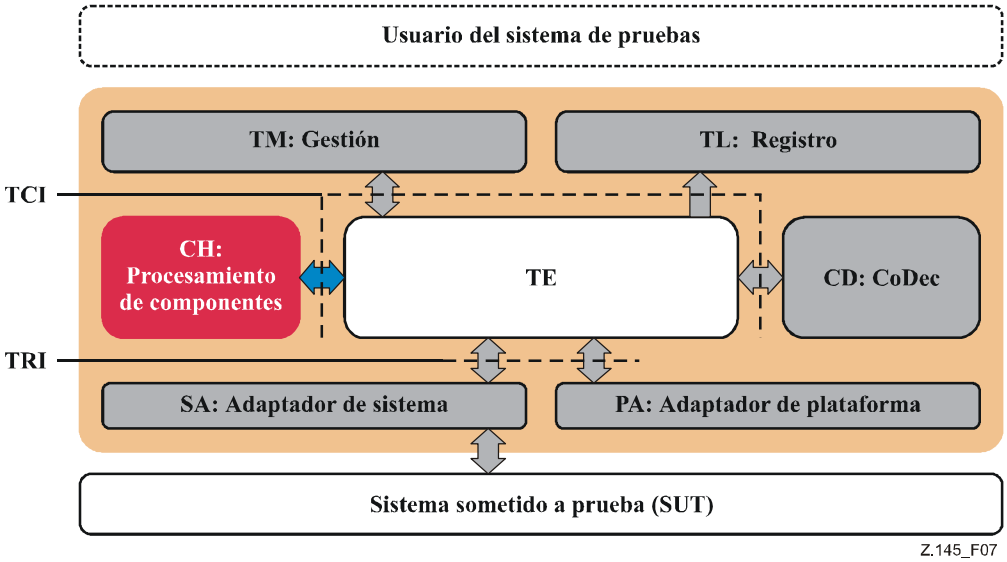

Figura 7/Z.145 – La interfaz TCI-CH

En una implementación de procesamiento de componentes se distribuyen, en un puerto conectado entre uno o varios TE que participan en una sesión de pruebas, diversas operaciones de configuración TTCN-3 como `create` (crear), `connect` (conectar) y `start` (iniciar), y de comunicaciones entre componentes como `send` (enviar). Cabe observar que, si bien en una sesión de pruebas pueden participar varios ejemplares de un TE, esto no es obligatorio.

El principio básico consiste en que la TCI-CH no *implementa* ningún tipo de funcionalidad TTCN-3. En su lugar, el TE le informa de que, por ejemplo, se ha de crear un componente de prueba. Basándose en la información que posee el procesamiento de componentes (CH, *component handling*), se transmitirá la solicitud de creación de un componente a

otro TE (distante) que participa. Este segundo TE creará el componente TTCN-3 y proporcionará un proceso al TE (local) solicitante, que podrá ahora funcionar en el componente creado a través de dicho proceso de componente.

En las definiciones de operaciones se emplean los términos "TE local" y "TE distante" para hacer énfasis en que una implementación de sistema de pruebas puede estar distribuida entre varios dispositivos, cada uno de los cuales contiene un TE completo. Las expresiones "local" y "distante" siempre se refieren a las interfaces que están siendo descritas en ese instante. A los efectos prácticos, "local" siempre será el TE que es llamado para una operación (en caso de operaciones *requeridas)* o que la llama (invoca) (en caso de operaciones *proporcionadas*). Mientras se considera que el TE es conceptualmente distribuido, el CH se supone no distribuido. Esto se puede lograr bien sea a través de una arquitectura centralizada o bien utilizando una plataforma de *middleware* que haga abstracción de los aspectos relativos a la distribución. Aunque el TE puede estar distribuido entre varios dispositivos físicos, puede haber configuraciones en las que participe en la sesión de pruebas solamente un TE no distribuido. En este caso, ambos términos, "local" y "distante", se refieren al mismo ejemplar de TE.

En la cláusula 10 se explica la utilización, por parte del TE o del CH, de las llamadas de operaciones y su ordenamiento secuencial.

Si bien todos los TE que participan en una sesión de pruebas son iguales, existe un TE* particular: aquél para el cual se han procesado la `tciStartTestCase`() o la `tciStartControl`() explícitas. Esto se debe a que el TE* ha de calcular el veredicto general. El TE* informará a la TM de la terminación de una ejecución de pruebas y proporcionará el veredicto general para el caso de pruebas.

7.3.3.1 Requeridas por la TCI-CH

En esta cláusula se especifican las operaciones que el CH requiere del TE. Además de las especificadas en esta subcláusula, son necesarias todas las operaciones *requeridas* por la interfaz TCI-CD.

7.3.3.1.1 tciEnqueueMsgConnected

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

Parámetros In (de entrada)	`Sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`Receiver`	Identificador del componente que recibe.
	`rcvdMessage`	El valor que se debe poner en la cola.
Valor devuelto	`Void`	
Restricción	El CH ha de invocar esta operación en el TE local, cuando se haya invocado en el TE distante una `tciSendConnected` *proporcionada*.	
Efecto	El TE pone el valor recibido en la cola de puerto local del componente receptor indicado.	

7.3.3.1.2 tciEnqueueCallConnected

Firma	`void tciEnqueueCallConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList)`	

Parámetros In (de entrada)	`Sender`	Identificador de puerto en el componente que envía, y a través del cual se envía el mensaje.
	`Receiver`	Identificador del componente que recibe.
	`Signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciCallConnected` *proporcionada.* Todos los parámetros de procedimiento *in* e *inout* contienen valores. Todos los parámetros de procedimiento *out* contienen el valor `null` puesto que sólo son importantes en una respuesta a la invocación del procedimiento mas no en su invocación propiamente dicha. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3.	
Efecto	El TE pone las invocaciones en la cola de puerto local del componente receptor indicado.	

7.3.3.1.3 tciEnqueueReplyConnected

Firma	`void tciEnqueueReplyConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList,`
	`in Value returnValue)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`receiver`	Identificador del componente que recibe la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
	`returnValue`	(Facultativo) devuelve el valor de la llamada de procedimiento.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciReplyConnected` *proporcionada*. Todos los parámetros de procedimiento *out* e *inout* y el valor devuelto contienen valores. Todos los parámetros de procedimiento *in* contienen el valor `null` puesto que sólo son importantes en la llamada de procedimiento mas no en la respuesta a ella. La `parameterList` contiene parámetros de llamada de procedimiento, que son especificados en la plantilla de firma TTCN-3. Si no se ha definido tipo de retorno para la firma de procedimiento en la ATS TTCN-3, se pasará el valor `null` como `returnValue`.	
Efecto	El TE pone la respuesta en la cola de puerto local del componente receptor indicado.	

7.3.3.1.4 tciEnqueueRaiseConnected

Firma	`void tciEnqueueRaiseConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in Value exception)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`receiver`	Identificador del componente que recibe la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`exception`	La excepción.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciRaiseConnected` *proporcionada*.	
Efecto	El TE pone la excepción en la cola de puerto local del componente receptor indicado.	

7.3.3.1.5 tciCreateTestComponent

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

Parámetros In (de entrada)	`kind`	El tipo de componente que se debe crear, ya sea `MTC`, `PTC` o `CONTROL`.
	`componentType`	Identificador del tipo de componente TTCN-3 que se debe crear.
	`name`	Nombre del componente que se debe crear.
Valor devuelto	Un `TriComponentIdType` para el componente creado.	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una *tciCreateTestComponentReq proporcionada*. `componentType` debe fijarse al valor `null` cuando se tenga que crear un componente de pruebas del tipo `control`. `name` debe fijarse al valor `null` si no se ha dado ningún nombre en la declaración de creación TTCN-3.	
Efecto	El TE crea un componente de pruebas TTCN-3 del `componentType` y devuelve una referencia `TriComponentIdType` al CH. El CH comunica la referencia al TE distante.	

7.3.3.1.6 tciStartTestComponent

Firma	`void tciStartTestComponent(in TriComponentIdType component, in TciBehaviourIdType behaviour, in TciParamaterListType parameterList)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe iniciar. Tiene que ver con un identificador creado anteriormente al invocar `tciCreateTestComponent`
	`behaviour`	Identificador del comportamiento que se debe iniciar en el componente.
	`parameterList`	Lista de valores, en la que cada uno de ellos define un parámetro de la lista definida en la declaración de función TTCN-3 de la función que se está iniciando. Los parámetros de `parameterList` se ordenan conforme a su aparición en la firma TTCN-3 del caso de pruebas. Si no hay parámetros para pasar, se hace pasar el valor `null` o una `parameterList` vacía, es decir una cuya longitud sea cero.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciStartTestComponentReq` *proporcionada*. Al permitirse solamente parámetros *in* para las funciones que están siendo inicializadas (Rec. UIT-T Z.140 [2]), `parameterList` contiene solamente este tipo de parámetros.	
Efecto	El TE iniciará el comportamiento indicado en el componente indicado.	

7.3.3.1.7 tciStopTestComponent

Firma	`void tciStopTestComponent(in TriComponentIdType component)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe detener.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciStopTestComponentReq` *proporcionada*.	
Efecto	El TE interrumpirá el comportamiento indicado en el componente indicado.	

7.3.3.1.8 tciConnect

Firma	`void tciConnect (in TriPortIdType fromPort, in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas del que hay que conectarse.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que conectarse.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciConnect` *proporcionada*.	
Efecto	El TE conectará entre sí los puertos indicados.	

7.3.3.1.9 tciDisconnect

Firma	`void tciDisconnect (in TriPortIdType fromPort, in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas que se debe desconectar.
	`toPort`	Identificador del puerto de componente de pruebas que se debe desconectar.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciDisconnect` *proporcionada*.	
Efecto	El TE desconectará los puertos indicados.	

7.3.3.1.10 tciMap

Firma	`void tciMap (in TriPortIdType fromPort, in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas desde el que se debe establecer la correspondencia.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que establecer la correspondencia.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciMapReq` *proporcionada*.	
Efecto	El TE hará corresponder entre sí los puertos indicados.	

7.3.3.1.11 tciUnmap

Firma	`void tciUnmap (in TriPortIdType fromPort, in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas al que hay que anular la correspondencia.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que anular la correspondencia.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciUnmapReq` *proporcionada*.	
Efecto	El TE anulará la correspondencia de los puertos indicados.	

7.3.3.1.12 tciTestComponentTerminated

Firma	`void tciTestComponentTerminated (in TriComponentIdType component, in VerdictValue verdict)`	
Parámetros In (de entrada)	`component`	Identificador del componente que ha terminado.
	`verdict`	Veredicto después de la terminación de componente.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciTestComponentTerminatedReq` *proporcionada*.	
Efecto	Se notifica al TE local de la terminación de la componente de pruebas indicada en un TE distante. Puesto que una función que se esté ejecutando en un componente de pruebas sólo puede tener parámetros *in* (Rec. UIT-T Z.140 [2]), la operación `tciTestComponentTerminated` no tiene un parámetro `parameterList`.	

7.3.3.1.13 tciTestComponentRunning

Firma	`TBoolean tciTestComponentRunning (in TriComponentIdType component)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe verificar si está funcionando.
Valor devuelto	`true` si el componente indicado está aún ejecutando un comportamiento de prueba, `false` de lo contrario.	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciTestComponentRunningReq` *proporcionada*.	
Efecto	El TE local establece si el componente indicado está ejecutando un comportamiento de prueba, en cuyo caso se devuelve `true`. En los demás casos, p. ej. cuando el componente de prueba haya terminado la ejecución, o el componente de prueba no haya empezado, etc., se devolverá `false`. Tras haber devuelto la respuesta la operación, el CH comunicará el valor al TE distante.	

7.3.3.1.14 tciTestComponentDone

Firma	`TBoolean tciTestComponentDone (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe verificar si ha sido realizado.
Valor devuelto	`true` si el componente indicado ha completado su comportamiento, `false` de lo contrario	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante una `tciTestComponentDoneReq` *proporcionada*.	
Efecto	El TE local establece si el componente indicado ha completado la ejecución de su comportamiento de prueba, en cuyo caso se devuelve `true`. En los demás casos, p. ej. cuando el componente de prueba no haya empezado, o sigue en ejecución, se devolverá `false`. Tras la devolución de la respuesta por la operación, el CH comunicará el valor al TE distante.	

7.3.3.1.15 tciGetMTC

Firma	`TriComponentIdType tciGetMTC()`
Valor devuelto	Un valor `TriComponentIdType` del MTC si el MTC se ejecuta en el TE local, de lo contrario `null`.
Restricción	El CH invocará esta operación en el TE local adecuado cuando se haya invocado en el TE distante una `tciGetMTCReq` *proporcionada*.
Efecto	El TE local establece si el MTC se está ejecutando en el TE local, en cuyo caso se devuelve el id de componente del MTC. De lo contrario se devuelve el valor `null`. la operación no tendrá ningún efecto en la ejecución del MTC. Tras la devolución de la respuesta por la operación, el CH transmitirá el valor al TE distante.

7.3.3.1.16 tciExecuteTestCase

Firma	`void tciExecuteTestCase (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`	
Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
	`tsiPortList`	Contiene todos los puertos que han sido declarados en la definición del componente de sistema para el caso de pruebas, es decir los puertos TSI. Si no se ha definido explícitamente un componente de sistema para el caso de pruebas, la `tsiPortList` contiene todos los puertos de comunicación del MTC. Los puertos de la `tsiPortList` se ordenarán conforme a cómo figuran en sus respectivas declaraciones de tipo de componente TTCN-3. Si no hay puertos para hacer pasar, se pasará el valor `null` o una `tsiPortList` vacía, es decir una cuya longitud sea cero.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local adecuado cuando se haya invocado en el TE distante `tciExecuteTestCaseReq` *proporcionada*.	
Efecto	El TE local establece si se deberían crear conexiones estáticas a la SUT e inicializar medios de comunicación para puertos TSI.	

7.3.3.1.17 tciReset

Firma	`void tciReset ()`
Valor devuelto	`void`
Restricción	El CH invocará esta operación en TE locales adecuados cuando se haya invocado en el TE distante `tciResetReq` *proporcionada*.
Efecto	El TE puede decidir tomar medidas para reinicializar localmente el sistema de pruebas.

7.3.3.1.18 tciKillTestComponent

Firma	`void tciKillTestComponent(in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe suprimir.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante `tciKillTestComponentReq` *proporcionada*.	
Efecto	El TE detiene el comportamiento en el componente indicado , si fuere necesario, y lo transfiere al estado suprimido (killed).	

7.3.3.1.19 tciTestComponentAlive

Firma	`TBoolean tciTestComponentAlive (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe verificar si está activo.
Valor devuelto	`true` si el componente indicado está activo, `false` de lo contrario.	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante `tciTestComponentAliveReq` *proporcionada*.	
Efecto	El TE local establece si el componente indicado está activo. Tras la devolución de la respuesta por la operación, el CH comunicará el valor al TE distante.	

7.3.3.1.20 tciTestComponentKilled

Firma	`TBoolean tciTestComponentKilled (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente del que se debe verificar si está siendo suprimido.
Valor devuelto	`true` si el componente indicado ha sido suprimido, `false` de lo contrario.	
Restricción	El CH invocará esta operación en el TE local cuando se haya invocado en el TE distante `tciTestComponentKilledReq` *proporcionada*.	
Efecto	El TE local establece si el componente indicado está en el estado suprimido, en cuyo caso se devolverá `true`. De lo contrario, se devolverá `false`. Tras la devolución de la respuesta por la operación, el CH comunicará el valor al TE distante.	

7.3.3.2 Proporcionadas por la TCI-CH

En esta cláusula se especifican las operaciones que el CH ha de proporcionar al TE.

7.3.3.2.1 tciSendConnected

Firma	`void tciSendConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in Value sendMessage)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`receiver`	Identificador del componente que recibe.
	`sendMessage`	El mensaje que se ha de enviar.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío unidifusión TTCN-3, en un puerto de componente que ha sido conectado a otro puerto de componente.	
Efecto	Envía una transmisión asíncrona solamente al componente receptor dado. El CH comunica al TE distante cuál receptor está siendo ejecutado y pone en cola la información en el TE distante.	

7.3.3.2.2 tciSendConnectedBC

Firma	`void tciSendConnectedBC (in TriPortIdType sender,` `in Value sendMessage)`	
Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`sendMessage`	El mensaje que se ha de enviar.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío por difusión TTCN-3 en un puerto de componente que ha sido conectado a otros puertos de componente.	
Efecto	Envía una transmisión asíncrona a todos los componentes receptores dados. El CH transmite a los TE distantes cuáles receptores están siendo ejecutados y pone en cola la información en los TE distantes.	

7.3.3.2.3 tciSendConnectedMC

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

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`receivers`	Identificadores de los componentes que reciben.
	`sendMessage`	El mensaje que se ha de enviar.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío multidifusión TTCN-3, en un puerto de componente que ha sido conectado a otros puertos de componente.	
Efecto	Envía una transmisión asíncrona a todos los componentes receptores dados. El CH transmite a los TE distantes cuáles receptores están siendo ejecutados y pone en cola la información en los TE distantes.	

7.3.3.2.4 tciCallConnected

Firma	`void tciCallConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`receiver`	Identificador del componente que recibe.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío unidifusión TTCN-3, en un puerto de componente que ha sido conectado a otro puerto de componente. Todos los parámetros de procedimiento *in* e *inout* contienen valores. Todos los parámetros de procedimiento *out* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento más no en su invocación propiamente dicha. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3.	
Efecto	Al invocar esta operación, el TE puede iniciar la llamada correspondiente de procedimiento del identificador de firma `signature` en el receptor de componente llamado (o invocado). La operación `tciCallConnected` devolverá el resultado sin esperar la conclusión de la llamada de procedimiento emitida. Obsérvese que no se incluye en la firma de operación `tciCallConnected` un valor facultativo de expiración de temporizador, que puede ser especificado en la ATS TCN-3 para una operación de llamada. El TE se encarga de ello mediante el inicio de un temporizador para la operación de llamada TTCN-3 en el PA, con llamada de operación TRI aparte, es decir `triStartTimer`. El CH transmite la llamada al TE distante, en el cual se está ejecutando `receiver` y pone la llamada en cola en el TE distante.	

7.3.3.2.5 tciCallConnectedBC

Firma	`void tciCallConnectedBC (in TriPortIdType sender,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío de difusión TTCN-3 en un puerto de componente que ha sido conectado a otros puertos de componente. Todos los parámetros de procedimiento *in* e *inout* contienen valores. Todos los parámetros de procedimiento *out* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento mas no en su invocación propiamente dicha. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3.	
Efecto	Al invocar esta operación, el TE puede iniciar la llamada correspondiente de procedimiento del identificador de firma `signature` en el receptor de componente llamado. La operación `tciCallConnected` devolverá el resultado sin esperar la conclusión de la llamada de procedimiento emitida. Obsérvese que no se incluye en la firma de operación `tciCallConnected` un valor facultativo de expiración de temporizador, que puede ser especificado en la ATS TCN-3 para una operación de llamada. El TE se encarga de ello mediante el inicio de un temporizador para la operación de llamada TTCN-3 en el PA, con llamada de operación TRI aparte, es decir `triStartTimer`. El CH transmite la llamada a todos los TE distantes, en los cuales se está ejecutando `receiver` y pone la llamada en cola en los TE distantes.	

7.3.3.2.6 tciCallConnectedMC

Firma	`void tciCallConnectedMC (in TriPortIdType sender,`
	`in TriComponentIdListType receivers,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`receivers`	Identificador del componente que recibe.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío multidifusión TTCN-3 en un puerto de componente que ha sido conectado a otros puertos de componente. Todos los parámetros de procedimiento *in* e *inout* contienen valores. Todos los parámetros de procedimiento *out* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento más no en su invocación propiamente dicha. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3.	
Efecto	Al invocar esta operación, el TE puede iniciar la llamada correspondiente de procedimiento del identificador de firma `signature` en el receptor de componente llamado. La operación `tciCallConnected` devolverá el resultado sin esperar la conclusión de la llamada de procedimiento emitida. Obsérvese que no se incluye en la firma de operación `tciCallConnected` un valor facultativo de expiración de temporizador, que puede ser especificado en la ATS TTCN-3 para una operación de llamada. El TE se encarga de ello mediante el inicio de un temporizador para la operación de llamada TTCN-3 en el PA, con llamada de operación TRI aparte, es decir `triStartTimer`. El CH transmite la llamada a todos los TE distantes, en los cuales se está ejecutando `receiver` y pone la llamada en cola en los TE distantes.	

7.3.3.2.7 tciReplyConnected

Firma	`void tciReplyConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList,`
	`in Value returnValue)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto en el componente que envía y a través del cual se envía el mensaje.
	`receiver`	Identificador del componente que recibe.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
	`returnValue`	(Opcional) devuelve el valor de la llamada de procedimiento.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de envío unidifusión TTCN-3 en un puerto de componente que ha sido conectado a otro puerto de componente. Todos los parámetros de procedimiento *out* e *inout* contienen valores. Todos los parámetros de procedimiento *in* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento más no en su invocación propiamente dicha. La `parameterList` contiene parámetros de procedimiento de llamada. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3. Si no se ha definido tipo de devolución para la firma de procedimiento en la ATS TTCN-3, se pasará el valor `null` como valor retornado.	
Efecto	Al invocar esta operación, el CH puede emitir la respuesta a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `receiver`. El CH transmite la respuesta al TE distante, en el cual se está ejecutando `receiver` y pone la respuesta en cola en el TE distante.	

7.3.3.2.8 tciReplyConnectedBC

Firma	`void tciReplyConnectedBC (in TriPortIdType sender,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList,`
	`in Value returnValue)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
	`returnValue`	(Opcional) devuelve el valor de la llamada de procedimiento.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de respuesta de difusión TTCN-3 en un puerto de componente que ha sido conectado a otro puerto de componente. Todos los parámetros de procedimiento *out* e *inout* contienen valores. Todos los parámetros de procedimiento *in* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento más no en su invocación propiamente dicha. La `parameterList` contiene parámetros de procedimiento de llamada. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3. Si no se ha definido tipo de devolución para la firma de procedimiento en la ATS TTCN-3, se pasará el valor `null` como valor retornado.	
Efecto	Al invocar esta operación, el CH puede emitir la respuesta a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `receiver`. El CH transmite la respuesta a los TE distantes, en los cuales se está ejecutando `receiver` y pone la respuesta en cola en los TE distantes.	

7.3.3.2.9 tciReplyConnectedMC

Firma	`void tciReplyConnectedMC (in TriPortIdType sender,`
	`in TriComponentIdListType receivers,`
	`in TriSignature IdType signature,`
	`in TciParameterListType parameterList,`
	`in Value returnValue)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`receivers`	Identificador de componentes que reciben la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`parameterList`	Lista de parámetros de valor que forman parte de la firma indicada. Los parámetros de `parameterList` se ordenan conforme a su aparición en la declaración de firma TTCN-3.
	`returnValue`	(Opcional) devuelve el valor de la llamada de procedimiento.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación de respuesta multidifusión TTCN-3 en un puerto de componente que ha sido conectado a otro puerto de componente. Todos los parámetros de procedimiento *out* e *inout* contienen valores. Todos los parámetros de procedimiento *in* contienen el valor `null`, puesto que sólo son importantes en una respuesta a la invocación del procedimiento más no en su invocación propiamente dicha. La `parameterList` contiene parámetros de procedimiento de llamada. Los parámetros de procedimiento son los especificados en la plantilla de firma TTCN-3. Si no se ha definido tipo de devolución para la firma de procedimiento en la ATS TTCN-3, se pasará el valor `null` como valor retornado.	
Efecto	Al invocar esta operación, el CH puede emitir la respuesta a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `receiver`. El CH transmite la respuesta a los TE distantes, en los cuales se está ejecutando `receiver` y pone la respuesta en cola en los TE distantes.	

7.3.3.2.10 tciRaiseConnected

Firma	`void tciRaiseConnected (in TriPortIdType sender,`
	`in TriComponentIdType receiver,`
	`in TriSignature IdType signature,`
	`in Value exception)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`receiver`	Identificador de componentes que reciben la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`exception`	El valor excepción.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación raise unidifusión TTCN-3 en un puerto de componente que ha sido conectado a otro puerto de componente.	
Efecto	Al invocar esta operación, el CH puede generar una excepción a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `receiver`. El CH transmite la excepción al TE distante, en el cual se está ejecutando `receiver` y pone la respuesta en cola en el TE distante.	

7.3.3.2.11 tciRaiseConnectedBC

Firma	`void tciRaiseConnectedBC (in TriPortIdType sender,`
	`in TriSignature IdType signature,`
	`in Value exception)`	

Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`exception`	El valor excepción.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación raise de difusión TTCN-3 en un puerto de componente que ha sido conectado a otros puertos de componente.	
Efecto	Al invocar esta operación, el CH puede generar una excepción a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `sender`. El CH transmite la excepción a los TE distantes, en los cuales se está ejecutando `receiver` y pone la respuesta en cola en los TE distantes.	

7.3.3.2.12 tciRaiseConnectedMC

Firma	`void tciRaiseConnectedMC (in TriPortIdType sender, in TriComponentIdListType receiver, in TriSignature IdType signature, in Value exception)`	
Parámetros In (de entrada)	`sender`	Identificador de puerto que envía la respuesta.
	`receivers`	Identificador de componentes que reciben la respuesta.
	`signature`	Identificador de la firma de la llamada de procedimiento.
	`exception`	El valor excepción.
Valor devuelto	`void`	
Restricción	El TE ha de invocar esta operación cuando ejecute una operación raise multidifusión TTCN-3 en un puerto de componente que ha sido conectado a otros puertos de componente.	
Efecto	Al invocar esta operación, el CH puede generar una excepción a una llamada de procedimiento correspondiente al identificador de firma `signature` y al identificador de componente `receiver`. El CH transmite la excepción a los TE distantes, en los cuales se está ejecutando `receiver` y pone la respuesta en cola en los TE distantes.	

7.3.3.2.13 tciCreateTestComponentReq

Firma	`TriComponentIdType tciCreateTestComponentReq (in TciTestComponentKindType kind, in Type componentType, in TString name)`	
Parámetros In (de entrada)	`kind`	El tipo de componente que se ha de crear, ya sea `MTC`, `PTC` o `CONTROL`.
	`componentType`	Identificador del tipo de componente TTCN-3 que se ha de crear.
Valor devuelto	Un valor `TriComponentIdType` para el componente creado.	
Restricción	El TE ha de invocar esta operación cuando haya que crear un componente, bien sea explícitamente cuando se invoque la operación create TTCN-3 o implícitamente cuando se deba crear el componente de prueba principal (MTC) o un componente de control. `name` se ha de fijar al valor `null` si no se ha dado nombre en la declaración de creación TTCN-3.	
Efecto	El CH transmite la creación de componente al TE distante e invoca en él la operación `tciCreateTestComponent` con el fin de obtener un identificador de componente para este componente.	

7.3.3.2.14 tciStartTestComponentReq

Firma	`void tciStartTestComponentReq(in TriComponentIdType component, in TciBehaviourIdType behaviour, in TciParamaterListType parameterList)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe iniciar.
	`behaviour`	Identificador del comportamiento que se debe iniciar en el componente.
	`parameterList`	Lista de `values`, en la que cada uno de ellos define un parámetro de la lista definida en la declaración de función TTCN-3 de la función que se está iniciando. Los parámetros de `parameterList` se ordenan conforme a su aparición en la firma TTCN-3 del caso de pruebas. Si no hay parámetros para pasar, se hace pasar el valor `null` o una `parameterList` vacía, es decir una cuya longitud sea cero.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación start TTCN-3. Al permitirse solamente parámetros *in* para las funciones que están siendo inicializadas (Rec. UIT-T Z.140 [2]), `parameterList` contiene solamente este tipo de parámetros.	
Efecto	El CH transmite la petición de componente start al TE distante e invoca en él la operación `tciStartTestComponent`.	

7.3.3.2.15 tciStopTestComponentReq

Firma	`void tciStopTestComponentReq(in TriComponentIdType component)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe detener.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación stop TTCN-3.	
Efecto	El CH transmite la petición de componente stop al TE distante e invoca en él la operación `tciStopTestComponent`.	

7.3.3.2.16 tciConnectReq

Firma	`void tciConnectReq (in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas del que hay que conectarse.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que conectarse.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación connect TTCN-3.	
Efecto	El CH transmite la petición de conexión al TE distante e invoca en él la operación `tciConnect` para establecer una conexión lógica entre los dos puertos indicados. Obsérvese que ambos puertos pueden estar en TE distantes, en cuyo caso, sólo se devuelve el resultado de la operación después de haber invocado en ambos TE distantes la operación `tciConnect`.	

7.3.3.2.17 tciDisconnectReq

Firma	`void tciDisconnectReq (in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas que hay que desconectar.
	`toPort`	Identificador del puerto de componente de pruebas que hay que desconectar.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación disconnect TTCN-3.	
Efecto	El CH transmite la petición de conexión al TE distante e invoca en él la operación `tciDisconnect` para interrumpir una conexión lógica entre los dos puertos indicados. Obsérvese que ambos puertos pueden estar en TE distantes, en cuyo caso, sólo se devuelve el resultado de la operación después de haber invocado en ambos TE distantes la operación `tciDisconnect`.	

7.3.3.2.18 tciMapReq

Firma	`void tciMapReq (in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas desde el que se debe establecer la correspondencia.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que establecer la correspondencia.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación map TTCN-3.	
Efecto	El CH transmite la petición de correspondencia al TE distante e invoca en él la operación `tciMap` para establecer una conexión lógica entre los dos puertos indicados.	

7.3.3.2.19 tciUnmapReq

Firma	`void tciUnmapReq (in TriPortIdType fromPort, in TriPortIdType toPort)`	
Parámetros In (de entrada)	`fromPort`	Identificador del puerto de componente de pruebas desde el que se debe interrumpir la correspondencia.
	`toPort`	Identificador del puerto de componente de pruebas al que hay que interrumpir la correspondencia.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación unmap TTCN-3.	
Efecto	El CH transmite la petición de anulación de correspondencia al TE distante e invoca en él la operación `tciUnmap` para interrumpir una conexión lógica entre los dos puertos indicados.	

7.3.3.2.20 tciTestComponentTerminatedReq

Firma	`void tciTestComponentTerminatedReq (in TriComponentIdType component, in VerdictValue verdict)`	
Parámetros In (de entrada)	`component`	Identificador del componente que ha terminado.
	`verdict`	Veredicto después de la terminación de componente.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando un componente de pruebas termine su ejecución, ya sea explícitamente mediante la operación stop TTNC-3 o implícitamente, si ha llegado a la última declaración.	
Efecto	Se notifica al CH local de la terminación del componente de pruebas indicado. Puesto que una función que se esté ejecutando en un componente de pruebas sólo puede tener parámetros *in* (Rec. UIT-T Z.140 [2]), la operación `tciTestComponentTerminateReq` no tiene un parámetro `parameterList`. El CH comunica la terminación del componente indicado a todos los TE involucrados y al TE* especial, que mantiene un récord del veredicto general.	

7.3.3.2.21 tciTestComponentRunningReq

Firma	`TBoolean tciTestComponentRunningReq (in TriComponentIdType component)`	
Parámetros In (de entrada)	`component`	Identificador del componente que se debe verificar si está funcionando.
Valor devuelto	`true` si el componente indicado está aún ejecutando un comportamiento de prueba, `false` de lo contrario.	
Restricción	El TE invocará esta operación cuando ejecute la operación running TTCN-3.	
Efecto	El CH transmite la petición de funcionamiento al TE distante que tiene el componente que se ha de verificar, e invoca en él la operación `tciTestComponentRunning` para verificar el estado de ejecución del componente de prueba indicado.	

7.3.3.2.22 tciTestComponentDoneReq

Firma	`TBoolean tciTestComponentDoneReq (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe verificar si ha sido realizado.
Valor devuelto	`true` si el componente indicado ha completado la ejecución de su comportamiento, `false` de lo contrario.	
Restricción	El TE invocará esta operación cuando ejecute la operación `done` TTCN-3.	
Efecto	El CH transmite la petición done al TE distante que tiene el componente que se ha de verificar, e invoca en él la operación `tciTestComponentDone` para verificar el estado de ejecución del componente de prueba indicado.	

7.3.3.2.23 tciGetMTCReq

Firma	`TriComponentIdType tciGetMTCReq()`
Valor devuelto	Un `TriComponentIdType` valor del MTC.
Restricción	El TE invocará esta operación cuando ejecute la operación `mtc` TTCN-3.
Efecto	El CH determina el componente del MTC.

7.3.3.2.24 tciExecuteTestCaseReq

Firma	`void tciExecuteTestCaseReq (in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`	
Parámetros In (de entrada)	`testCaseId`	Un identificador de caso de prueba definido en el módulo TTCN-3.
	`tsiPortList`	`tsiPortList` contiene todos los puertos que han sido declarados en la definición del componente de sistema para el caso de pruebas, es decir los puertos TSI. Si no se ha definido explícitamente un componente de sistema para el caso de pruebas, la `tsiPortList` contiene todos los puertos de comunicación del MTC. Los puertos de la `tsiPortList` se ordenarán conforme a cómo figuran en sus respectivas declaraciones de tipo de componente TTCN-3. Si no hay puertos para hacer pasar, se pasará el valor `null` o una `tsiPortList` vacía, es decir una cuya longitud sea cero.
Valor devuelto	`void`	
Restricción	El TE puede invocar esta operación justo antes de iniciar el comportamiento de caso de pruebas en el MTC (durante el transcurso de una operación execute TTCN-3).	
Efecto	El CH transmite la petición de ejecución del caso de pruebas a los TE distantes que tienen puertos de sistema en el caso de pruebas indicado. Se pueden configurar conexiones estáticas al SUT e inicializar los medios de comunicación.	

7.3.3.2.25 tciResetReq

Firma	`void tciResetReq ()`
Valor devuelto	`void`
Restricción	El TE puede invocar esta operación en cualquier momento para reinicializar el sistema.
Efecto	El CH transmite la petición de reinicialización a los TE en cuestión.

7.3.3.2.26 tciKillTestComponentReq

Firma	`void tciKillTestComponentReq(in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe suprimir.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación cuando ejecute la operación `kill` TTCN-3.	
Efecto	El CH transmite la petición de supresión de componente a los TE distantes e invoca en ellos la operación `tciKillTestComponent`.	

7.3.3.2.27 tciTestComponentAliveReq

Firma	`TBoolean tciTestComponentAliveReq (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe verificar si está activo.
Valor devuelto	`true` si el componente indicado está activo, `false` de lo contrario.	
Restricción	El TE invocará esta operación cuando ejecute la operación `alive` TTCN-3.	
Efecto	El CH transmite la petición al TE distante que ha creado el componente de prueba en cuestión, e invoca en él la operación `tciTestComponentAlive` para verificar el estado del componente de prueba indicado.	

7.3.3.2.28 tciTestComponentKilledReq

Firma	`TBoolean tciTestComponentKilledReq (in TriComponentIdType comp)`	
Parámetros In (de entrada)	`comp`	Identificador del componente que se debe verificar si está siendo suprimido.
Valor devuelto	`true` si el componente indicado ha sido suprimido, `false` de lo contrario.	
Restricción	El TE invocará esta operación cuando ejecute la operación `killed` TTCN-3.	
Efecto	El CH transmite la petición al TE distante que ha creado el componente de prueba en cuestión, e invoca en él la operación `tciTestComponentKilled` para verificar el estado del componente de prueba indicado.	

7.3.4 La interfaz TCI-TL

La interfaz de registro de pruebas TCI (TCI-TL) describe las operaciones que debe implementar un ejecutable TTCN-3 (TE) y las que una implementación de registro de pruebas ha de proporcionar al TE (figura 8).

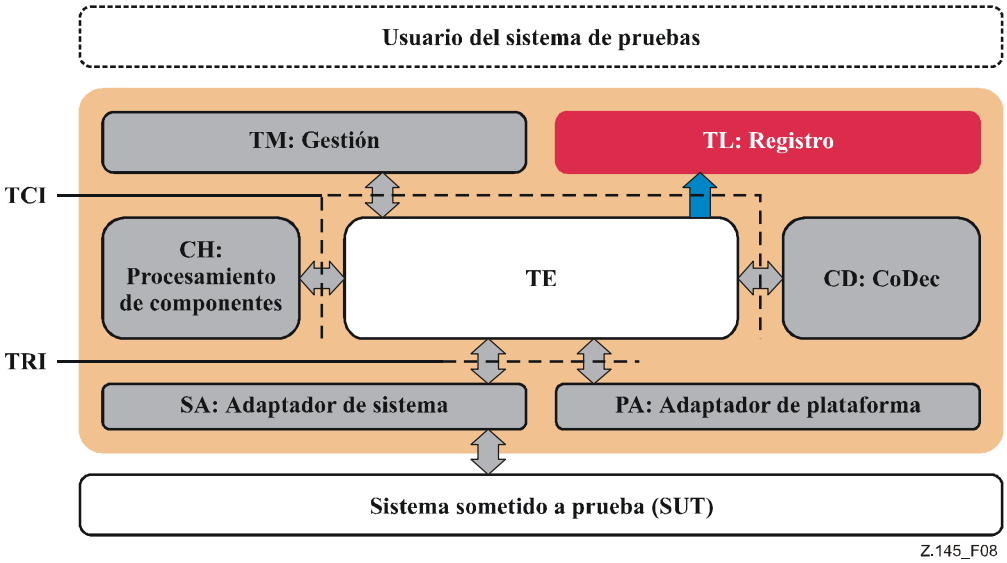

Figura 8/Z.145 – La interfaz TCI-TL

La funcionalidad de registro permite a toda operación de nivel TTCN-3 registrar para el usuario el evento que están realizando el TE, el SA, el PA, el CH o la CD.

7.3.4.1 Proporcionadas por la TCI-TL

En esta cláusula se especifican las operaciones que el TL ha de proporcionar al TE.

7.3.4.1.1 tliTcExecute

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`tcId`	El caso de prueba que se ha de ejecutar.
	`pars`	La lista de parámetros requeridos por caso de prueba.
	`dur`	Duración de la ejecución.
Valor devuelto	`void`	
Restricción	El TE debe invocar esta operación para obtener un registro de la petición de ejecución de caso de prueba.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.2 tliTcStart

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`tcId`	El caso de prueba que se ha de ejecutar.
	`pars`	La lista de parámetros requeridos por caso de prueba.
	`dur`	Duración de la ejecución.
Valor devuelto	`void`	
Restricción	El TE debe invocar esta operación para registrar el inicio de un caso de prueba. Este evento ocurre antes del inicio del caso de pruebas.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.3 tliTcStop

Firma	`void tliTcStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la interrupción de un caso de prueba.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.4 tliTcStarted

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`tcid`	El caso de prueba que se ha de ejecutar.
	`pars`	La lista de parámetros requeridos por caso de prueba.
	`dur`	Duración de la ejecución.
Valor devuelto	`void`	
Restricción	El TM invocará esta operación para registrar el inicio de un caso de prueba. Este evento ocurre después del inicio del caso de pruebas.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.5 tliTcTerminated

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`tcId`	El caso de prueba que se ha de ejecutar.
	`pars`	La lista de parámetros requeridos por caso de prueba.
	`outcome`	El veredicto del caso de prueba.
Valor devuelto	`void`	
Restricción	El TM invocará esta operación para registrar la terminación de un caso de prueba. Este evento ocurre después de la terminación del caso de pruebas.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.6 tliCtrlStart

Firma	`void tliCtrlStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TM invocará esta operación para registrar el inicio de la parte de control. Este evento ocurre antes del inicio de la parte de control. Si el control no está representado por un componente TRI, c es null.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.7 tliCtrlStop

Firma	`void tliCtrlStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la interrupción de la parte de control. Este evento ocurre antes de la interrupción de la parte de control. Si el control no está representado por un componente TRI, c es null.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.8 tliCtrlTerminated

Firma	`void tliCtrlTerminated (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TM invocará esta operación para registrar la terminación de la parte de control. Este evento ocurre después de la terminación de la parte de control. Si el control no está representado por un componente TRI, c es null.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.9 tliMSend_m

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`address`	La dirección del destino en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`msg`	El mensaje codificado.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de envío unidifusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.10 tliMSend_m_BC

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`msg`	El mensaje codificado.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de envío de difusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.11 tliMSend_m_MC

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`addresses`	Las direcciones de los destinos en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`msg`	El mensaje codificado.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de envío multidifusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.12 tliMSend_c

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`to`	El componente que recibirá el mensaje.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de envío unidifusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.13 tliMSend_c_BC

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de envío de difusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.14 tliMSend_c_MC

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`msgValue`	El valor que se ha de codificar y enviar.
	`toList`	Los componentes que recibirán el mensaje.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de envío multidifusión. Este evento ocurre después del envío. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.15 tliMDetected_m

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msg`	El mensaje codificado recibido.
	`address`	La dirección de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la puesta en cola de un mensaje. Este evento ocurre después de la puesta en cola del mensaje. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.16 tliMDetected_c

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msgValue`	El mensaje recibido.
	`from`	El componente que envío el mensaje.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar la puesta en cola de un mensaje. Este evento ocurre después de la puesta en cola del mensaje. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.17 tliMMismatch_m

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msgValue`	El mensaje que se compara con el modelo (plantilla).
	`msgTmpl`	Plantilla empleada para verificar la correspondencia de mensaje.
	`diffs`	Diferencia o falta de correspondencia entre plantilla y mensaje
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una plantilla. Este evento ocurre después de verificar la correspondencia de plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.18 tliMMismatch_c

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msgValue`	El mensaje que se compara con el modelo (plantilla).
	`msgTmpl`	La plantilla empleada para verificar la correspondencia de mensaje.
	`diffs`	Diferencia o falta de correspondencia entre plantilla y mensaje
	`from`	El componente que envío el mensaje.
	`fromTmpl`	El componente de usuario esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una plantilla. Este evento ocurre después de verificar la correspondencia de plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.19 tliMReceive_m

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msgValue`	El mensaje que se compara con el modelo (plantilla).
	`msgTmpl`	Plantilla empleada para verificar la correspondencia de mensaje.
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la recepción de un mensaje. Este evento ocurre después de verificar la correspondencia de plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.20 tliMReceive_c

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe el mensaje.
	`msgValue`	El mensaje que se compara con el modelo (plantilla).
	`msgTmpl`	Plantilla empleada para verificar la correspondencia de mensaje.
	`from`	El componente que envío el mensaje.
	`fromTmpl`	El componente de usuario esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la recepción de un mensaje. Este evento ocurre después de verificar la correspondencia de plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.21 tliPrCall_m

Firma	`void tliPrCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`address`	La dirección del destino en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`pars`	Los parámetros codificados.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de llamada unidifusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.22 tliPrCall_m_BC

Firma	`void tliPrCall_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`pars`	Los parámetros codificados.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de llamada de difusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.23 tliPrCall_m_MC

Firma	`void tliPrCall_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`addresses`	Las direcciones de los destinos en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`pars`	Los parámetros codificados.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de llamada multidifusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.24 tliPrCall_c

Firma	`void tliPrCall_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriComponentIdType to, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`to`	El componente que recibirá el mensaje.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de llamada unidifusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.25 tliPrCall_c_BC

Firma	`void tliPrCall_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de llamada de difusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.26 tliPrCall_c_MC

Firma	`void tliPrCall_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se invoca la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros de la operación llamada.
	`tohist`	El componente que recibirá el mensaje.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de llamada multidifusión. Este evento ocurre después de la ejecución de llamada. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.27 tliPrGetCallDetected_m

Firma	`void tliPrGetCallDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TriParameterListType pars, in TriAddressType address)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la llamada.
	`signature`	La firma de la llamada detectada.
	`pars`	Los parámetros codificados de la llamada detectada.
	`address`	La dirección del destino en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación de puesta en cola `getcall`. Este evento ocurre después de la puesta en cola de la llamada. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga es algo que está fuera del alcance de esta Recomendación.	

7.3.4.1.28 tliPrGetCallDetected_c

Firma	`void tliPrGetCallDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TriComponentIdType from)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe la llamada.
	`signature`	La firma de la operación llamada.
	`parsValue`	Los parámetros codificados de la llamada detectada.
	`from`	El componente que invocó la operación.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar la operación de puesta en cola `getcall`. Este evento ocurre después de la puesta en cola de la llamada. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.29 tliPrGetCallMismatch_m

Firma	`void tliPrGetCallMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe la llamada.
	`signature`	La firma de la llamada detectada.
	`parsValue`	Los parámetros de la llamada detectada.
	`parsTmpl`	La plantilla empleada para verificar la correspondencia de parámetro.
	`diffs`	La diferencia o falta de correspondencia entre la llamada y la plantilla
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación de puesta en cola getcall. Este evento ocurre después de la comparación de getcall con una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.30 tliPrGetCallMismatch_c

Firma	`void tliPrGetCallMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe la llamada.
	`signature`	La firma de la llamada detectada.
	`parsValue`	Los parámetros de la llamada detectada.
	`parsTmpl`	La plantilla empleada para verificar la correspondencia de parámetro.
	`diffs`	Diferencia o falta de correspondencia entre plantilla y mensaje
	`from`	El componente que invocó la operación.
	`fromTmpl`	El componente que llama esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación de puesta en cola getcall. Este evento ocurre después de la comparación de getcall con una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.31 tliPrGetCall_m

Firma	`void tliPrGetCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe la llamada.
	`signature`	La firma de la llamada detectada.
	`parsValue`	Los parámetros de la llamada detectada.
	`parsTmpl`	La plantilla empleada para verificar la correspondencia de parámetro.
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación de puesta en cola getcall. Este evento ocurre después de que se ha establecido la correspondencia entre getcall y una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.32 tliPrGetCall_c

Firma	`void tliPrGetCall_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto en que se recibe la llamada.
	`signature`	La firma de la llamada detectada.
	`parsValue`	Los parámetros de la llamada detectada.
	`parsTmpl`	La plantilla empleada para verificar la correspondencia de parámetro.
	`from`	El componente que invocó la operación.
	`fromTmpl`	El componente que llama esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación de puesta en cola getcall. Este evento ocurre después de que se ha establecido la correspondencia entre getcall y una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.33 tliPrReply_m

Firma	`void tliPrReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`address`	La dirección del destino en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`repl`	La respuesta codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de respuesta unidifusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.34 tliPrReply_m_BC

Firma	`void tliPrReply_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`repl`	La respuesta codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de respuesta de difusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.35 tliPrReply_m_MC

Firma	`void tliPrReply_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`addresses`	Las direcciones de los destinos en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`repl`	La respuesta codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación de respuesta multidifusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.36 tliPrReply_c

Firma	`void tliPrReply_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriComponentIdType to, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`to`	El componente que recibe la respuesta.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de respuesta unidifusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.37 tliPrReply_c_BC

Firma	`void tliPrReply_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de respuesta de difusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.38 tliPrReply_c_MC

Firma	`void tliPrReply_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value parsValue, in Value replValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía el mensaje.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta que se ha de enviar.
	`toList`	Los componentes que recibirán la respuesta.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación de respuesta multidifusión. Este evento ocurre después de la ejecución de la respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.39 tliPrGetReplyDetected_m

Firma	`void tliPrGetReplyDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TriParameterType repl, in TriAddressType address)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`repl`	La respuesta codificada recibida.
	`address`	La dirección de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación de puesta en cola getreply. Este evento ocurre después de que getreply se ha puesto en cola. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.40 tliPrGetReplyDetected_c

Firma	`void tliPrGetReplyDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value replValue, in TriComponentIdType from)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`replValue`	La respuesta recibida.
	`from`	El componente que envió la respuesta.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación de puesta en cola getreply. Este evento ocurre después de que getreply se ha puesto en cola. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.41 tliPrGetReplyMismatch_m

Firma	`void tliPrGetReplyMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta recibida.
	`replyTmpl`	La plantilla utilizada para verificar la correspondencia de la respuesta.
	`diffs`	La diferencia o falta de correspondencia entre la plantilla y la respuesta
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación getreply. Este evento ocurre después de que se haya comparado getreply con una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.42 tliPrGetReplyMismatch_c

Firma	`void tliPrGetReplyMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta recibida.
	`replyTmpl`	La plantilla utilizada para verificar la correspondencia de la respuesta.
	`diffs`	La diferencia o falta de correspondencia entre la plantilla y la respuesta
	`from`	El componente que envió la respuesta.
	`fromTmpl`	El componente que se espera responda.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación getreply. Este evento ocurre después de que se haya comparado getreply con una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.43 tliPrGetReply_m

Firma	`void tliPrGetReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta recibida.
	`replyTmpl`	La plantilla utilizada para verificar la correspondencia de la respuesta.
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la obtención (*getting*) de una respuesta. Este evento ocurre después de que se haya comparado getreply con una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.44 tliPrGetReply_c

Firma	`void tliPrGetReply_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la respuesta.
	`signature`	La firma relacionada con la respuesta.
	`parsValue`	Los parámetros de firma relacionados con la respuesta.
	`replValue`	La respuesta recibida.
	`replyTmpl`	La plantilla utilizada para verificar la correspondencia de la respuesta.
	`from`	El componente que envió la respuesta.
	`fromTmpl`	El componente que responde esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la obtención de una respuesta. Este evento ocurre después de que se haya comparado getreply con una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.45 tliPrRaise_m

Firma	`void tliPrRaise_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressType address, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`address`	La dirección del destino en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`exc`	La excepción codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación raise unidifusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.46 tliPrRaise_m_BC

Firma	`void tliPrRaise_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`exc`	La excepción codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación raise de difusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.47 tliPrRaise_m_MC

Firma	`void tliPrRaise_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`addresses`	Las direcciones de los destinos en el SUT.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`exc`	La excepción codificada.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar una operación raise multidifusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.48 tliPrRaise_c

Firma	`void tliPrRaise_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdType to, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`to`	El componente que recibe la respuesta.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación raise unidifusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.49 tliPrRaise_c_BC

Firma	`void tliPrRaise_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación raise de difusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.50 tliPrRaise_c_MC

Firma	`void tliPrRaise_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se envía la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción que se ha de enviar.
	`toList`	Los componentes que recibirán la respuesta.
	`transmissionFailure`	El mensaje de fallo que puede ocurrir en la transmisión.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar una operación raise multidifusión. Este evento ocurre después de la ejecución de una respuesta. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.51 tliPrCatchDetected_m

Firma	`void tliPrCatchDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TriExceptionType exc, in TriAddressType address)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`exc`	La excepción atrapada.
	`address`	La dirección de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación de puesta en cola catch. Este evento ocurre después de que catch ha sido puesta en cola. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.52 tliPrCatchDetected_c

Firma	`void tliPrCatchDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in Value excValue, in TriAddressType address)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`excValue`	La excepción atrapada.
	`address`	La dirección de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación de puesta en cola catch. Este evento ocurre después de que catch ha sido puesta en cola. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.53 tliPrCatchMismatch m

Firma	`void tliPrCatchMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción recibida.
	`excTmpl`	La plantilla utilizada para verificar la correspondencia de la excepción.
	`diffs`	La diferencia o falta de correspondencia entre la plantilla y la excepción
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación catch. Este evento ocurre después de que catch ha sido comparada con una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.54 tliPrCatchMismatch_c

Firma	`void tliPrCatchMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción recibida.
	`excTmpl`	La plantilla utilizada para verificar la correspondencia de la excepción.
	`diffs`	La diferencia o falta de correspondencia entre la plantilla y la excepción
	`from`	El componente que envió la respuesta.
	`fromTmpl`	El componente que responde esperado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación catch. Este evento ocurre después de que catch ha sido comparada con una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.55 tliPrCatch_m

Firma	`void tliPrCatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción recibida.
	`excTmpl`	La plantilla utilizada para verificar la correspondencia de la excepción.
	`address`	La dirección de la fuente en el SUT.
	`addressTmpl`	La dirección esperada de la fuente en el SUT.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la obtención de una excepción. Este evento ocurre después de que catch ha sido comparada con una plantilla. Se utiliza para registrar la comunicación con el SUT.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.56 tliPrCatch_c

Firma	`void tliPrCatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
	`parsValue`	Los parámetros de firma relacionados con la excepción.
	`excValue`	La excepción recibida.
	`excTmpl`	La plantilla utilizada para verificar la correspondencia de la excepción.
	`from`	El componente que envió la respuesta.
	`fromTmpl`	El componente que responde esperado.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar la obtención de una excepción. Este evento ocurre después de que catch ha sido comparada con una plantilla. Se utiliza para registrar la comunicación entre componentes.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.57 tliPrCatchTimeoutDetected

Firma	`void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar la detección de una expiración de temporizador catch. Este evento ocurre después de que se ha puesto en cola la expiración del temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.58 tliPrCatchTimeout

Firma	`void tliPrCatchTimeout (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignature IdType signature)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto a través del cual se recibe la excepción.
	`signature`	La firma relacionada con la excepción.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que se alcanza la expiración de temporizador. Este evento ocurre después de que se ha alcanzado la expiración del temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.59 tliCCreate

Firma	`void tliCCreate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TString name)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se está creando.
	`name`	El nombre del componente que se está creando.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación crear componente. Este evento ocurre después de la creación de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.60 tliCStart

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se está iniciando.
	`beh`	El comportamiento que se está iniciando en el componente.
	`pars`	Los parámetros del comportamiento iniciado.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación inicio de componente. Este evento ocurre después del inicio de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.61 tliCRunning

Firma	`void tliCRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se verifica si está funcionando.
	`status`	El estado de este componente.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación funcionamiento de componente. Este evento ocurre después del funcionamiento de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.62 tliCAlive

Firma	`void tliCAlive(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se verifica si está funcionando.
	`status`	El estado de este componente.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación componente en activo. Este evento ocurre después de componente en activo.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.63 tliCStop

Firma	`void tliCStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se ha detenido.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación interrupción de componente. Este evento ocurre después de interrupción de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.64 tliCKill

Firma	`void tliCKill(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`comp`	El componente que se ha detenido.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación destrucción de componente. Este evento ocurre después de la destrucción de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.65 tliCDoneMismatch

Firma	`void tliCDoneMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`compTmpl`	La plantilla utilizada para verificar la correspondencia de done.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación de componente done. Este evento ocurre después de que se haya comparado done con una plantilla.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.66 tliCDone

Firma	`void tliCDone (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`compTmpl`	La plantilla utilizada para verificar la correspondencia de done.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación de componente done. Este evento ocurre después de done.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.67 tliCKilledMismatch

Firma	`void tliCKilledMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`compTmpl`	La plantilla utilizada para verificar la correspondencia de done.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una operación suprimida. Este evento ocurre después de que se compara la supresión con una plantilla.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.68 tliCKilled

Firma	`void tliCKilled (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`compTmpl`	La plantilla utilizada para verificar la correspondencia de done.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación terminación de componente. Este evento ocurre después de la terminación de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.69 tliCTerminated

Firma	`void tliCTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`verdict`	El veredicto del componente.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación terminación de componente. Este evento ocurre después de la terminación de componente.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.70 tliPConnect

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`c1`	El componente del primer puerto que hay que conectar.
	`port1`	El primer puerto que hay que conectar.
	`c2`	El componente del segundo puerto que hay que conectar.
	`port2`	El segundo puerto que hay que conectar.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar la operación conectar. Este evento ocurre después de la operación conectar.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.71 tliPDisconnect

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`c1`	El componente del primer puerto que hay que desconectar.
	`port1`	El primer puerto que hay que desconectar.
	`c2`	El componente del segundo puerto que hay que desconectar.
	`port2`	El segundo puerto que hay que desconectar.
Valor devuelto	`void`	
Restricción	El CH invocará esta operación para registrar la operación desconectar. Este evento ocurre después de la operación desconectar.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.72 tliPMap

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`c1`	El componente del primer puerto al que hay efectuar la correspondencia.
	`port1`	El primer puerto al que hay efectuar la correspondencia.
	`c2`	El componente del segundo puerto al que hay efectuar la correspondencia.
	`port2`	El segundo puerto al que hay efectuar la correspondencia.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación map. Este evento ocurre después de la operación map.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.73 tliPUnmap

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`c1`	El componente del primer puerto al que hay anular la correspondencia.
	`port1`	El primer puerto al que hay anular la correspondencia.
	`c2`	El componente del segundo puerto al que hay anular la correspondencia.
	`port2`	El segundo puerto al que hay anular la correspondencia.
Valor devuelto	`void`	
Restricción	El SA invocará esta operación para registrar la operación unmap. Este evento ocurre después de la operación unmap.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.74 tliPClear

Firma	`void tliPClear(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto que se debe liberar.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación liberación de puerto. Este evento ocurre después de la operación liberación de puerto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.75 tliPStart

Firma	`void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto que se debe inicializar.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación inicio de puerto. Este evento ocurre después de la operación inicio de puerto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.76 tliPStop

Firma	`void tliPStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto que se debe interrumpir.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación parada de puerto. Este evento ocurre después de la operación parada de puerto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.77 tliPHalt

Firma	`void tliPHalt(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`port`	El puerto que se debe.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación parada de puerto. Este evento ocurre después de la operación parada de puerto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.78 tliEncode

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`value`	El valor que se ha de codificar.
	`encoderFailure`	El mensaje de fallo que puede ocurrir en la codificación.
	`msg`	El valor codificado.
	`codec`	El codificador utilizado.
Valor devuelto	`void`	
Restricción	El CD invocará esta operación para registrar la operación codificar.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.79 tliDecode

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`msg`	El valor que se ha de decodificar.
	`decoderFailure`	El mensaje de fallo que puede ocurrir en la decodificación.
	`value`	El valor decodificado.
	`codec`	El decodificador utilizado.
Valor devuelto	`void`	
Restricción	El CD invocará esta operación para registrar la operación decodificar.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.80 tliTTimeoutDetected

Firma	`void tliTTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timer`	El temporizador que ha expirado.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar la operación detección de la expiración de un temporizador. Este evento ocurre después de que se haya puesto en cola la expiración de un temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.81 tliTTimeoutMismatch

Firma	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timerTmpl`	La plantilla de temporizador que no corresponde.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de la expiración de un temporizador. Este evento ocurre después de que haya fallado la correspondencia de la expiración de un temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.82 tliTTimeoutMismatch

Firma	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timerTmpl`	La plantilla de temporizador que corresponde.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la correspondencia de la expiración de un temporizador. Este evento ocurre después de la correspondencia de la expiración de un temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.83 tliTStart

Firma	`void tliTStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timer`	El temporizador que ha empezado.
	`dur`	La duración del temporizador.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar el inicio de un temporizador. Este evento ocurre después de la operación inicio de temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.84 tliTStop

Firma	`void tliTStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timer`	El temporizador que se ha parado.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar la interrupción de un temporizador. Este evento ocurre después de la operación interrupción de temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.85 tliTRead

Firma	`void tliTRead(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType elapsed)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timer`	El temporizador que ha empezado.
	`elapsed`	La duración del temporizador.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar la lectura de un temporizador. Este evento ocurre después de la operación lectura de temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.86 tliTRunning

Firma	`void tliTRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`timer`	El temporizador que se verifica si está funcionando.
	`status`	El estado de este componente.
Valor devuelto	`void`	
Restricción	El PA invocará esta operación para registrar el funcionamiento de un temporizador. Este evento ocurre después de la operación funcionamiento de temporizador.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.87 tliSEnter

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`name`	El nombre del ámbito.
	`parsValue`	Los parámetros del ámbito.
	`kind`	El tipo de ámbito.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la entrada en un alcance (*scope*). Este evento ocurre después de que se haya entrado al alcance.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.88 tliSLeave

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`name`	El nombre del alcance.
	`val`	El valor devuelto del alcance.
	`kind`	El tipo de alcance.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la salida de un alcance (*scope*). Este evento ocurre después de que se haya salido del alcance.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.89 tliVar

Firma	`void tliVar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value varValue)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`name`	El nombre de la variable.
	`varValue`	El nuevo valor de la variable.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la modificación del valor de una variable. Este evento ocurre después de que se han modificado los valores.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.90 tliModulePar

Firma	`void tliModulePar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value parValue)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`name`	El nombre del parámetro de módulo.
	`parValue`	El valor del parámetro de módulo.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar el valor de un parámetro de módulo. Este evento ocurre después del acceso al valor de un parámetro de módulo.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.91 tliGetVerdict

Firma	`void tliGetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`verdict`	El valor actual del veredicto local.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación getverdict. Este evento ocurre después de la operación getverdict.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.92 tliSetVerdict

Firma	`void tliSetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`verdict`	El valor que se ha de poner al veredicto local.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la operación setverdict. Este evento ocurre después de la operación setverdict.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.93 tliLog

Firma	`void tliLog (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciValueList log)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`log`	La cadena que se debe almacenar en el registro.
Valor devuelto	`void`	
Restricción	La TM invocará esta operación para el registro de declaraciones TTCN-3. Este evento ocurre después de la operación log TTCN-3.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.94 tliAEnter

Firma	`void tliAEnter(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que se está entrando en una alt. Este evento ocurre después de que se haya entrado en una alt.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.95 tliALeave

Firma	`void tliALeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que se está abandonando una alt. Este evento ocurre después de que se haya abandonado una alt.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.96 tliANomatch

Firma	`void tliANomatch (in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la falta de correspondencia de una alt. Este evento ocurre después de que se haya establecido la falta de correspondencia de una alt.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.97 tliARepeat

Firma	`void tliARepeat(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que se está repitiendo una alt. Este evento ocurre después de que se haya repetido una alt.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.98 tliADefaults

Firma	`void tliADefaults(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que se está entrando a la sección por defecto. Este evento ocurre después de que se haya entrado a una sección por defecto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.99 tliAActivate

Firma	`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)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`name`	El nombre del defecto.
	`pars`	El parámetro del defecto.
	`ref`	La referencia de defecto resultante.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la activación de un defecto. Este evento ocurre después de que se haya activado un defecto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.100 tliADeactivate

Firma	`void tliADeactivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value ref)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
	`ref`	La referencia de defecto resultante.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar la desactivación de un defecto. Este evento ocurre después de que se haya desactivado un defecto.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

7.3.4.1.101 tliAWait

Firma	`void tliAWait(in string am, in long ts, in string src, in long line, in TriComponentId c)`	
Parámetros In (de entrada)	`am`	Un mensaje adicional.
	`ts`	La hora en la que ocurre el evento.
	`src`	El fichero fuente de la especificación de prueba.
	`line`	El número de línea en que se realiza la petición.
	`c`	El componente que produce este evento.
Valor devuelto	`void`	
Restricción	El TE invocará esta operación para registrar que el componente está esperando eventos para una nueva instantánea (*snapshot)*.	
Efecto	El TL presenta al usuario toda la información proporcionada en los parámetros de esta operación, aunque cómo lo haga está fuera del alcance de esta Recomendación.	

8 Correspondencia de lenguaje Java

8.1 Introducción

En esta cláusula se introduce la correspondencia de lenguaje Java TCI. En aras de la eficiencia, se emplea una correspondencia de lenguaje específica en lugar del IDL del OMG para el lenguaje Java.

En la correspondencia de lenguaje Java para la interfaz de control TTCN-3 (TCI) se indica cómo hacer corresponder las definiciones IDL descritas en la cláusula 7 con el lenguaje Java. Puesto que sólo se utilizan construcciones básicas Java, la correspondencia de lenguaje es independiente de la versión de Java que se esté empleando.

8.2 Nombres y alcances

8.2.1 Nombres

Aunque no hay conflictos entre los identificadores utilizados en la definición del IDL y el lenguaje Java, se aplican a los identificadores IDL algunas reglas de traducción de nombres.

En las interfaces Java o en los identificadores de clase se omite el `Type` que va al final en la definición IDL.

EJEMPLO: el tipo IDL **`TciTestCaseIdType`** corresponde en Java a **`TciTestCaseId`**.

La correspondencia así obtenida es conforme a los convenios normalizados de codificación para Java.

8.2.2 Alcances

Se hace corresponder el módulo IDL **`tciInterface`** al paquete Java `org.etsi.ttcn3.tci`. Todas las declaraciones de tipo IDL en este módulo se hacen corresponder a clases Java o a declaraciones de interfaz en dicho paquete.

Correspondencia de tipo

Correspondencia básica de tipo

En el cuadro 2 se indica la correspondencia entre los tipos nativos TBoolean, TFloat, TInteger, TString y TStringSeq y los tipos Java.

Cuadro 2/Z.145 – Correspondencias básicas de tipo

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

El tipo nativo TObjId se define en la cláusula correspondiente a la interfaz ObjidValue.

boolean

El tipo IDL **`TBoolean`** corresponde al tipo básico Java `boolean`.

float

El tipo IDL **`TFloat`** corresponde al tipo básico Java `float`.

char

El tipo IDL **`TChar`** corresponde al tipo básico Java `char`.

int

El tipo IDL **`TInteger`** corresponde al tipo básico Java `int`.

String

El tipo IDL **`TString`** corresponde a la clase Java `java.lang.String` sin verificación de intervalo o límites para los caracteres de la cadena. Todas las posibles cadenas definidas en TTCN-3 se pueden convertir a `java.lang.String`.

String[]

El tipo IDL **`TStringSeq`** corresponde a un arreglo de la clase `java.lang.String`.

Universal Char

El tipo IDL **`TUniversalChar`** corresponde al tipo básico Java `int`. El entero emplea la forma canónica definida en 6.2/X.290.

Correspondencia de tipo estructurado

En la descripción IDL TCI se definen como tipos nativos los tipos definidos por el usuario. En la correspondencia de lenguaje Java estos tipos se hacen corresponder a interfaces Java. La interfaz define los métodos y atributos que tienen a su disposición los objetos que la implementan.

8.2.2.1 TciParameterType

`TciParameterType` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getParameterName ()` Devuelve el nombre de parámetro definido en la especificación TTCN-3;
- `setParameterName (String name)` Fija el nombre del parámetro `TciParameter` a `name`;
- `getParameterPassingMode()` Devuelve el modo de paso de este parámetro;
- `setParameterPassingMode(TciParameterPassingMode mode)` Fija el modo del parámetro `TriParameter` a `mode`;
- `getParameter()` Devuelve el parámetro Value de este TciParameter, o el objeto `null` si el parámetro contiene el valor `null`;
- `setParameter(Value parameter)` Fija el parámetro Value de este TciParameter a `parameter`. Si se ha de poner `null` para indicar que este parámetro no tiene valor, se hará pasar el `null` Java como parámetro.

8.2.2.2 TciParameterPassingModeType

`TciParameterPassingModeType` corresponde a la siguiente interfaz:

```
// 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;
}
```

Constantes:

- `TCI_IN` Sirve para indicar que un `TciParameter` es un parámetro `in`;
- `TCI_INOUT` Sirve para indicar que un `TciParameter` es un parámetro `inout`;
- `TCI_OUT` Sirve para indicar que un `TciParameter` es un parámetro `out`.

8.2.2.3 TciParameterListType

`TciParameterListType` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `size()` Devuelve el número de parámetros en la lista;
- `isEmpty()` Devuelve `true` si la lista no contiene parámetros;
- `getParameters()` Devuelve una `Enumeration` de los parámetros de la lista. La enumeración entrega los parámetros en el mismo orden que en la lista;
- `get(int index)` Devuelve el `TciParameter` en la posición especificada;
- `clear()` Suprime todos los parámetros de la lista `TciParameterList`;
- `add(TciParameter parameter)` Añade `parameter` al final de esta `TciParameterList`;
- `setParameter(TciParameter[] parameters)` Rellena esta `TciParameterList con parameters`.

8.2.2.4 TciTypeClassType

`TciTypeClassType` corresponde a la siguiente interfaz:

```
// 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` corresponde a la siguiente interfaz:

```
// 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` corresponde a la siguiente interfaz:

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

8.2.2.7 TciTestCaseIdType

`TciTestCaseIdType` corresponde a la siguiente interfaz:

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

8.2.2.8 TciModuleIdType

`TciModuleIdType` corresponde a la siguiente interfaz:

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

8.2.2.9 TciModuleParameterIdType

`TciModuleParameterIdType` corresponde a la siguiente interfaz:

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

8.2.2.10 TciModuleParameterListType

`TciModuleParameterListType` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `size()` Devuelve el número de parámetros de módulo en la lista;
- `isEmpty()` Devuelve `true` si la lista no contiene parámetros de módulo;

- `getModuleParameters()` Devuelve una `Enumeration` de los parámetros de módulo en la lista. La enumeración entrega los parámetros de módulo en el mismo orden que en la lista;
- `get(int index)` Devuelve el `TciModuleParameter` en la posición especificada.

8.2.2.11 TciModuleParameterType

`TciModuleParameterType` corresponde a la siguiente interfaz:

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

Métodos:

- `getModuleParameterName ()` Devuelve el nombre de parámetro de módulo definido en la especificación TTCN-3;
- `getDefaultValue()` Devuelve el valor por defecto de este `TciModuleParameter`, o el objeto `null` si el módulo contiene el valor `null`.

8.2.2.12 TciParameterTypeListType

`TciParameterTypeListType` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `size()` Devuelve la cantidad de tipos de parámetro en la lista;
- `isEmpty()` Devuelve `true` si la lista no contiene tipos de parámetro;
- `getParameterTypes()` Devuelve una `Enumeration` de los tipos de parámetros en la lista. La enumeración entrega los tipos de parámetros en el mismo orden que en la lista;
- `get(int index)` Devuelve el `TciParameterType` en la posición especificada.

8.2.2.13 TciModuleIdListType

`TciModuleIdListType` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `size()` Devuelve la cantidad de módulos en esta lista;
- `isEmpty()` Devuelve `true` si esta lista no contiene módulos;
- `getImportedModules()` Devuelve una `Enumeration` de los módulos en la lista. La enumeración entrega los módulos en el mismo orden que en la lista;
- `get(int index)` Devuelve el `TciModuleId` en la posición especificada.

8.2.3 Correspondencia de tipo abstracto

Los tipos de información TTCN-3 se modelan en Java mediante la correspondencia de tipo abstracto que se define en la presente cláusula. La interfaz `Type` define solamente operaciones empleadas para recuperar tipos definidos TTCN-3. No se pueden construir tipos TTCN-3 utilizando la interfaz `Type`. Los modelos de los tipos se obtienen gracias a la interfaz `Type`, que proporciona métodos para identificar tipos y obtener valores de un determinado tipo.

8.2.3.1 Tipo (`Type`)

`Type` corresponde a la siguiente interfaz:

```
// 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();
}
```

Métodos:

- `getDefiningModule()` Devuelve el identificador del módulo en el que se ha definido el tipo. Si el tipo representa un tipo base TTCN-3 se devuelve el valor `null`;
- `getName()` Devuelve el nombre del tipo definido en el módulo TTCN-3;
- `getTypeClass()` Devuelve la clase del tipo respectivo. `TciTypeClassType` puede ser una de las siguientes constantes: `ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OBJID`, `OCTETSTRING`, `RECORD`, `RECORD_OF`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`, `VERDICT`;
- `newInstance()` Devuelve un valor reciente de un tipo determinado. Este valor inicial del valor creado es indefinido;
- `getTypeEncoding()` Devuelve el atributo de codificación de tipo definido en el módulo TTCN-3;
- `getTypeEncodingVariant()` Devuelve el atributo variante de codificación de valor definido en TTCN-3, si lo hubiere. Si no se ha definido atributo variante de codificación, se devuelve el valor `null`;
- `getTypeEncoding()` Devuelve el atributo de extensión de tipo definido en el módulo TTCN-3.

8.2.4 Correspondencia de valor abstracto

Es posible obtener los valores TTCN-3 del TE y construirlos utilizando la interfaz Value. La interfaz de correspondencia de valores se construye jerárquicamente y Value es la interfaz básica. Se han definido interfaces especializadas para los diversos tipos de valores.

8.2.4.1 Valor (`Value`)

`Value` corresponde a la siguiente interfaz:

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

Métodos:

- `getType()` Devuelve el tipo del valor especificado;
- `notPresent()` Devuelve `true` si el valor especificado es `omit`, `false` de lo contrario;

- `getValueEncoding()` Devuelve el atributo de codificación de valor definido en TTCN-3, si lo hubiere. Si no se ha definido atributo de codificación, se devuelve el valor `null`;
- `getValueEncodingVariant()` Devuelve el atributo variante de codificación definido en TTCN-3, si lo hubiere. Si no se ha definido atributo variante de codificación, se devuelve el valor `null`.

8.2.4.2 IntegerValue

El tipo **`IntegerValue`** corresponde a la siguiente interfaz:

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

Métodos:

- `setInteger(int value)` Fija este IntegerValue al valor entero `value`;
- `getInteger()` Devuelve el valor entero representado por este IntegerValue.

8.2.4.3 FloatValue

El tipo **`FloatValue`** corresponde a la siguiente interfaz:

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

Métodos:

- `setFloat(float value)` Fija este FloatValue al valor de punto flotante `value`;
- `getFloat()` Devuelve el valor de punto flotante representado por este FloatValue.

8.2.4.4 BooleanValue

El tipo **`BooleanValue`** corresponde a la siguiente interfaz:

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

Métodos:

- `setBoolean(boolean value)` Fija este BooleanValue al valor booleano `value`;
- `getBoolean()` Devuelve el valor booleano representado por este `BooleanValue`.

8.2.4.5 ObjidValue

`ObjidValue` corresponde a la siguiente interfaz:

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

Métodos:

- `getObjid()` Devuelve el valor de identificador de objeto TTCN-3 `objid`;
- `setObjid(TciObjId value)` Fija este `ObjidValue` a `value`.

8.2.4.6 TciObjId

`TciObjId` corresponde a la siguiente interfaz. La representación nativa en Java de un TTCN-3 consta de una secuencia ordenada de varios `TciObjIdElement`.

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

Métodos:

- `size()` Devuelve el tamaño de este Id de objeto en TciObjIdElements;
- `setObjElement(TciObjIdElement[] objElements)` Fija este ObjId conforme a la lista de objElements;
- `getObjElement(int index)` Devuelve el TciObjIdElement en la posición `index`.

8.2.4.7 TciObjIdElement

`TciObjIdElement` representa un solo elemento de objeto dentro de un valor ObjId TTCN-3. Se puede configurar utilizando diversas representaciones, por ejemplo la ASCII, o como entero.

`TciObjIdElement` corresponde a la siguiente interfaz:

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

Métodos:

- `setElementAsAscii(String element)` Fija la representación interna de este ObjIdElement al valor de cadena `element`;
- `setElementAsNumber(int element)` Fija la representación interna de este ObjIdElement al valor entero `element`;
- `getElementAsAscii()` Devuelve la representación interna de este ObjIdElement en forma de cadena;
- `getElementAsNumber()` Devuelve la representación interna de este ObjIdElement en forma de entero.

8.2.4.8 CharstringValue

`CharstringValue` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getString()` — Devuelve la cadena `charstring` TTCN-3. La representación textual de la `charstring` TTCN-3 es `"`, y su longitud es cero;
- `setString(String value)` — Fija la `CharstringValue` a `value`;
- `getChar(int position)` — Devuelve el valor de caracteres de la charstring TTCN-3 en `position`. La posición 0 indica el primer carácter de la `charstring` TTCN-3. Se aceptan valores de posición desde `0` hasta `length` - 1;
- `setChar(int position, char value)` — Fija el carácter en `position` a `value`. Se aceptan valores de `position` desde `0` hasta `length` - 1;
- `getLength()` — Devuelve la longitud en caracteres de este `CharstringValue`, es cero si el valor de `CharstringValue` es `omit`;
- `setLength(int len)` — Fija la longitud en caracteres de este `CharstringValue` a `len`.

8.2.4.9 BitstringValue

`BitstringValue` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getString()` — Devuelve la representación textual de `BitstringValue`, definida en TTCN-3. Por ejemplo, la representación textual de `0101` es `"0101"B`. La representación textual de la `bitstring` TTCN-3 vacía es `""B`, y su longitud, cero;
- `setString(String value)` — Fija el valor de `BitstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de `BitstringValue` será `0101` si la representación textual en value es `"0101"B`;
- `getBit(int position)` — Devuelve el valor `(0 | 1)` en position de esta cadena de bits TTCN-3. La posición 0 indica el primer bit de la bitstring TTCN-3. Se aceptan valores de posición desde `0` hasta `length - 1`;
- `setBit(int position, int value)` — Fija el bit en position al valor `(0 | 1)`. La posición 0 indica el primer bit de este BitstringValue. Se aceptan valores de position desde `0` hasta `length - 1`;
- `getLength()` — Devuelve la longitud en bits de este `BitstringValue`, es cero si el valor de este `BitstringValue` es `omit`;
- `setLength(int len)` — Fija la longitud en bits de este `BitstringValue` a `len`.

8.2.4.10 OctetstringValue

`OctetstringValue` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getString()` — Devuelve la representación textual de `OctetstringValue`, definida en TTCN-3. Por ejemplo, la representación textual de `0xCAFFEE` es `"CAFFEE"O`. La representación textual de la `octetstring` TTCN-3 vacía es `""O`, y su longitud, cero;
- `setString(String value)` — Fija el valor de `OctetstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de `OctetstringValue` será `0xCAFFEE` si la representación textual es `"CAFFEE"O`;
- `getOctet(int position)` — Devuelve el valor (0..255) en `position` de esta octetstring TTCN-3. La posición 0 indica el primer octeto de la octetstring TTCN-3. Se aceptan valores de posición desde `0` hasta `length - 1`;
- `setOctet(int position, int value)` — Fija el octeto en `position` al valor (0..255). La posición 0 indica el primer octeto de la octetstring. Se aceptan valores de posición desde `0` hasta `length - 1`;
- `getLength()` — Devuelve la longitud en octetos de esta `OctetstringValue`, es cero si el valor de esta `OctetstringValue` es `omit`;
- `setLength(int len)` — Fija la longitud en octetos de esta `OctetstringValue` a `len`.

8.2.4.11 UniversalCharstringValue

`UniversalCharstringValue` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getString()` — Devuelve la representación textual de `UniversalCharstringValue`, definida en TTCN-3;
- `setString(String value)` — Fija el valor de `UniversalCharstringValue` conforme a la representación textual definida por `value`;
- `getChar(int position)` — Devuelve el valor de `UniversalChar` de la `universal charstring` TTCN-3 en position. La posición 0 indica el primer `UniversalChar` de la `universal charstring` TTCN-3. Se aceptan valores de posición desde `0` hasta `length - 1`;

- `setChar(int position, char value)` — Fija el `UniversalChar` en `position` a `value`. Se aceptan valores de posición desde `0` hasta `length – 1`;
- `getLength()` — Devuelve la longitud de esta `UniversalCharstringValue` en `UniversalChars`, es cero si el valor de esta `UniversalCharstringValue` es `omit`;
- `setLength(int len)` — Fija la longitud de esta `UniversalCharstringValue` en `UniversalChars` a `len`.

8.2.4.12 HexstringValue

`HexstringValue` corresponde a la siguiente interfaz:

```
// 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);
}
```

Métodos:

- `getString()` — Devuelve la representación textual de `HextstringValue`, definida en TTCN-3. Por ejemplo, la representación textual de `0xAFFEE` es `"AFFEE"H`. La representación textual de la `hexstring` TTCN-3 vacía es `""H` y su longitud, cero;
- `setString(String value)` — Fija el valor de `HexstringValue` conforme a la representación textual definida por `value`. Por ejemplo, el valor de esta `HexstringValue` será `0xAFFEE` si la representación textual en `value` es `"AFFEE"H`;
- `getHex(int position)` — Devuelve el valor `(0..15)` en `position` de esta hexstring TTCN-3. La posición 0 indica el primer dígito hex de la hexstring TTCN-3. Se aceptan valores de posición desde `0` hasta `length - 1` ;
- `setHex(int position, int value)` — Fija el valor del dígito hex en position a `(0..16)`. La posición 0 indica el primer octeto en la `hexstring`. Se aceptan valores de posición desde `0` hasta `length - 1`;
- `getLength()` — Devuelve la longitud en octetos de `HexstringValue`, es cero si el valor de `HexstringValue` es `omit`;
- `setLength(int len)` — Fija la longitud en dígitos hex de `HexstringValue` a `len`.

8.2.4.13 RecordValue

`RecordValue` corresponde a la siguiente interfaz:

```
// 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() ;
}
```

Métodos:

- `getField(String fieldName)` — Devuelve el valor del campo `fieldName`. El valor devuelto es el `Value` del tipo básico abstracto común, puesto que un campo de registro puede tener cualquier tipo definido en TTCN-3. Si no se puede obtener el campo a partir del registro, se devuelve el valor `null`;

- `setField(String fieldName, Value value)` Fija el valor del campo denominado `fieldName` del registro a `value`. No se supondrá nada acerca de cómo se almacenan los campos en un registro. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido copiado. Se espera, entonces, que las modificaciones posteriores de `value` no serán reflejadas en el registro;
- `getFieldNames()` Devuelve un arreglo de cadena de cadena de nombres de campo, y el valor `null` si el registro no tiene campos.

8.2.4.14 RecordOfValue

`RecordOfValue` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `getField(String fieldName)` Devuelve el valor del `record of` en `position` si `position` está entre `zero` y `length - 1`, de lo contrario el valor `null`. El valor devuelto es el `Value` del tipo básico abstracto común, puesto que un `record of` puede tener cualquier tipo definido en TTCN-3;
- `setField(int position, Value value)` Fija el campo en `position` a `value`. Si `position` es mayor que `(length - 1)` se ampliará el `record of` para que incluya `(position + 1)`. Los elementos `record of` entre la posición original en `length` y `position - 1` se pondrán a `OMIT`. No se supondrá nada acerca de cómo se almacenan los campos en un `record of`. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido copiado. Se espera, entonces, que las modificaciones posteriores de `value` no serán consideradas en el `record of`;
- `appendField(Value value)` Añade el valor al final del `record of`, es decir en la posición `length`. No se supondrá nada acerca de cómo se almacenan los campos en un `record of`. Es posible que en una implementación interna se decida guardar una referencia a este valor o copiar el valor propiamente dicho. Conviene suponer que el valor ha sido copiado. Se espera, entonces, que las modificaciones posteriores de `value` no serán reflejadas en el `record of`;
- `getElementType()` Esta operación devuelve el `Type` de los elementos de este `record of`;
- `getLength()` Devuelve la longitud real del valor `record of`, es cero si el valor `record of` es `OMIT`;
- `setLength(int len)` Fija la longitud de `record of` a `len`. Si `len` es mayor que la longitud original, es porque hay elementos recién creados que tienen el valor `OMIT`. Si `len` es menor o igual que la longitud original, se omite esta operación.

8.2.4.15 UnionValue

`UnionValue` corresponde a la siguiente interfaz:

```
// 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 ();
}
```

Métodos:

- `getVariant(String variantName)` — Devuelve el valor de la unión TTCN-3 `variantName` si `variantName` es igual al resultado de `getPresentVariantName`, o `null` de lo contrario `variantName` indica el nombre de la variante de unión, definido en TTCN-3;
- `setVariant(String variantName, Value value)` — Fija el `variantName` de la unión a value. Si no se define `variantName` para esta unión, se ignorará esta operación. Si se escogió otra variante, se seleccionará en su lugar la nueva;
- `getPresentVariantName()` — Devuelve el nombre de la variante de unión cuyo valor haya sido puesto a `String`. Si no se ha escogido variante, se devolverá el valor `null`;
- `getVariantNames()` — Devuelve un arreglo de `String` de nombres de variantes, el valor `null` si la unión no tiene campos. Si el `UnionValue` representa el `anytype` TTCN-3, es decir que la clase del tipo obtenido por getType() es ANYTYPE, se devolverán todos los tipos TTCN-3 predefinidos y definidos por el usuario.

8.2.4.16 EnumeratedValue

`EnumeratedValue` corresponde a la siguiente interfaz:

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

Métodos:

- `getEnum()` — Devuelve el identificador de cadena de este `EnumeratedValue`. Este identificador es igual al de la especificación TTCN-3;
- `setEnum(String enumValue)` — Fija enum a `enumValue`. Si `enumValue` no es un valor permitido para esta enumeración, se ignorará la operación.

8.2.4.17 VerdictValue

`VerdictValue` corresponde a la siguiente interfaz:

```
// 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) ;
}
```

Métodos:

- `getverdict ()` Devuelve el valor entero de este `VerdictValue`. El `integer` es una de las siguientes constantes: `ERROR, FAIL, INCONC, NONE, PASS`;
- `setVerdict(int verdict)` Fija este `VerdictValue` a `verdict`. Cabe observar que siempre es posible fijar `VerdictValue` a cualquiera de dichos valores. El `VerdictValue` no efectúa ningún cálculo de veredicto, como se define en TTCN-3. Por ejemplo, es válido fijar primero `VerdictValue` a `ERROR` y luego a `PASS`.

8.2.4.18 AddressValue

`AddressValue` corresponde a la siguiente interfaz:

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

Métodos:

- `getAddress()` Devuelve el valor representado por `AddressValue`;
- `setAddress(Value value)` Fija `AddressValue` a `value`.

8.2.5 Correspondencia de tipos abstractos de registro

Para facilitar el registro de correspondencias entre valores y plantillas, se definen otros tipos.

8.2.5.1 TciValueTemplate

`TciValueTemplate` corresponde a la siguiente interfaz:

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

Métodos:

- `isOmit()` Devuelve `true` si la plantilla (o modelo) es `omit`, o `false` de lo contrario;
- `isAny()` Devuelve `true` si la plantilla es `any`, o `false` de lo contrario;
- `isAnyOrOmit()` Devuelve `true` si la plantilla es `anyoromit`, o `false` de lo contrario;
- `getTemplateDef()` Esta operación devuelve la definición de plantilla.

8.2.5.2 TciNonValueTemplate

`TciNonValueTemplate` corresponde a la siguiente interfaz:

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

Métodos:

- `isAny()` Devuelve `true` si la plantilla es `any`, o `false` de lo contrario;
- `isAll()` Devuelve `true` si la plantilla es `all`, o `false` de lo contrario;
- `getTemplateDef()` Esta operación devuelve la definición de plantilla.

8.2.5.3 TciValueList

`TciValueList` corresponde a la siguiente interfaz:

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

Métodos:

- `size()` Devuelve la cantidad de valores en esta lista;
- `isEmpty()` Devuelve `true` si la lista no contiene valores;
- `get(int index)` Devuelve el `Value` en la posición especificada.

8.2.5.4 TciValueDifference

`TciValueDifference` corresponde a la siguiente interfaz:

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

Métodos:

- `getValue()` Devuelve el valor de `TciValueDifference`;
- `getTciValueTemplate ()` Devuelve la plantilla de `TciValueDifference`;
- `getDescription()` Devuelve la descripción de la falta de correspondencia.

8.2.5.5 TciValueDifferenceList

`TciValueDifferenceList` corresponde a la siguiente interfaz:

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

Métodos:

- `size()` Devuelve la cantidad de diferencias en esta lista;
- `isEmpty()` Devuelve `true` si la lista no contiene diferencias;
- `get(int index)` Devuelve el `TciValueDifference` en la posición especificada.

8.3 Constantes

En esta correspondencia de lenguaje Java se han especificado constantes, todas las cuales se definen como `public final static` y es posible acceder a ellas desde cualquier objeto en cualquier paquete. Las constantes que se definen en esta cláusula no se definen en la correspondiente al IDL, sino que resultan de la especificación de los tipos IDL TCI marcados como nativos.

Es posible utilizar las siguientes constantes para establecer el modo de paso de parámetros TTCN-3 (véase también 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`.

En el caso del valor de parámetro `null`, se pondrá el valor de parámetro codificado al `null` de Java.

Se han de utilizar las siguientes constantes en el procesamiento de valores (véase también 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`.

Se han de utilizar las siguientes constantes en el procesamiento de componentes (véase también 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 Correspondencia de interfaces

En la definición IDL TCI se especifican cuatro interfaces, a saber la **`TCI-TM`**, la **`TCI-CH`**, la **`TCI-CD`** y la **`TCI-TL`**. Las operaciones se definen para diversos sentidos en dichas interfaces, es decir, algunas operaciones pueden ser invocadas solamente por el ejecutable TTCN-3 (TE), el adaptador de sistema (SA) o el adaptador de plataforma (PA) en el TMC,

mientras que otras pueden ser invocadas solamente por el TMC en el TE. Es por ello que ha sido necesario dividir las interfaces IDL TCI en dos subinterfaces, indicadas mediante `Required` o `Provided`.

Cuadro 3/Z.145 – Sub interfaces

Invoca/ invocada	TE	TM	CD	CH	SA	PA	TL
TE	–	Proporcionada por TCI-TM	Proporcio-nada por TCI-CD	Proporcionada por TCI-CH	Véase la Rec. UIT-T Z.144 [1]	Véase la Rec. UIT-T Z.144 [1]	Proporcionada por TCI-TL
TM	Requerida por TCI-TM	–	–	–	–	–	Proporcionada por TCI-TL
CD	Requerida por TCI-CD	–	–	–	–	–	Proporcionada por TCI-TL
CH	Requerida por TCI-CH	–	–	–	–	–	Proporcionada por TCI-TL
SA	Véase la Rec. UIT-T Z.144				–	–	Proporcionada por TCI-TL
PA	Véase la Rec. UIT-T Z.144				–	–	Proporcionada por TCI-TL
TL	–	–	–	–	–	–	–

Todos los métodos definidos en estas interfaces deben comportarse conforme a lo definido en la cláusula 7.

8.4.1 La interfaz TCI-TM

La interfaz **`TCI-TM`** se divide en dos subinterfaces, la `TCI-TM Provided`, que define llamadas (invocaciones) del TE al TM, y la `TCI-TM Required`, que define llamadas del TM al TE.

8.4.1.1 Proporcionada por la TCI-TM

`TCI-TM Provided` corresponde a la siguiente interfaz:

```
// 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 Requerida por la TCI-TM

`TCI-TM Required` corresponde a la siguiente interfaz:

```
// 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 La interfaz TCI-CD

La interfaz **`TCI-CD`** se divide en dos subinterfaces, la `TCI-CD Provided`, que define llamadas del TE al CD, y la `TCI-CD Required`, que define llamadas del CD al TE.

8.4.2.1 Proporcionada por la TCI-CD

`TCI-CD Provided` corresponde a la siguiente interfaz:

```
// 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 Requerida por la TCI-CD

`TCI-CD Required` corresponde a la siguiente interfaz:

```
// 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 La interfaz TCI-CH

La interfaz **`TCI-CH`** se divide en dos subinterfaces, la `TCI-CH Provided`, que define llamadas del TE al CH, y la `TCI-CH Required`, que define llamadas del CH al TE.

8.4.3.1 Proporcionada por la TCI-CH

`TCI-CH Provided` corresponde a la siguiente interfaz:

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

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

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

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

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

    public void     tciStartTestComponentReq(TriComponentId comp,
                                             TciBehaviourId 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 Requerida por la TCI-CH

`TCI-CH Required` corresponde a la siguiente interfaz:

```
// 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 La interfaz TCI-TL

La interfaz **TCI-TL** contiene solamente las subinterfaces TCI-TL Provided, que definen llamadas del TE, el TM, el CH, el CD, el SA y el PA al TL.

8.4.4.1 Proporcionada por la TCI-TL

TCI-TL Provided corresponde a la siguiente interfaz:

```
// 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 Parámetros facultativos

En la cláusula 7.3 se indica que se tiene que utilizar un valor reservado para indicar la ausencia de un parámetro facultativo. En el caso de la correspondencia de lenguaje Java, se empleará para ello el valor `null`. Así, por ejemplo, si se ha de ignorar en la operación `tciReplyConnected` el parámetro de valor, se debe invocar la operación `tciReplyConnected (sender, receiver, signature, parameterList, null)`.

8.6 Inicialización de la TCI

Todos los métodos son no estáticos, en otras palabras, sólo se pueden invocar las operaciones en objetos. Puesto que la presente Recomendación no define estrategias concretas de implementación para el TE, el TM, el CD y el CH, queda fuera de su alcance la forma como éstos aprenden el procesamiento de los respectivos objetos.

Los fabricantes de equipos deben proporcionar a los programadores de TM, CD y CH métodos para indicar éstos a quien corresponda.

8.7 Procesamiento de error

En todas las operaciones invocadas desde el TM, el CH o el CD se debe comunicar que se ha producido un resultado fructuoso. De haber identificado el TE un error, se comunicará al usuario un caso de prueba mediante el procedimiento definido en 7.3.1.2.5 (`tciError`). Si una operación invocada por el TE en el TM, el CH, el CD, o el TL provoca un error, esta situación debe comunicarse al TE utilizando los procedimientos definidos en 7.3.2.1.12 (`tciErrorReq`).

No se define ningún otro mecanismo de tratamiento de error en esta correspondencia de lenguaje Java. En particular, no se definen mecanismos de tratamiento de excepción.

9 Correspondencia de lenguaje ANSI-C

9.1 Introducción

En esta cláusula se define la correspondencia de lenguaje ANSI-C TRI para la información TCI especificada en 7.2 y las operaciones TCI especificadas en 7.3.

9.2 Interfaces de valor

Interfaz IDL TCI	Representación ANSI-C	Notas y comentarios
Tipo		
TciModuleIdType getDefiningModule()	`TciModuleIdType tciGetDefiningModule(TciType inst)`	
TString getName()	`String tciGetName(TciType inst)`	Tipo cadena reutilizado del IDL (recomendación del OMG)
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)`	
Valor		
TString getValueEncoding()	`String tciGetValueEncoding(TciValue inst)`	
TString getValueEncodingVariant()	`String tciGetValueEncodingVariant(TciValue inst)`	
Type getType()	`TciType tciGetType(TciValue inst)`	
TBoolean notPresent()	`Boolean tciNotPresent(TciValue inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
IntegerValue		
TInteger getInt()	`String tciGetIntAbs(TciValue inst)`	Devuelve el valor absoluto del entero (base 10) como cadena ASCII.
	`unsigned long int tciGetIntNumberOfDigits (TciValue inst)`	Devuelve la cantidad de cifras de un valor entero.
	`Boolean tciGetIntSign(TciValue inst)`	Devuelve true si el número es positivo, o false de lo contrario

Interfaz IDL TCI	Representación ANSI-C	Notas y comentarios
	`char tciGetIntDigit (TciValue inst, unsigned long int position)`	Devuelve el valor de la cifra en la posición 'position', donde la posición 0 indica la cifra menos significativa.
void setInt(in TInteger value)	`void tciSetIntAbs(TciValue inst, String value)`	Fija el valor absoluto del entero (base 10) a través de una cadena ASCII. La primera cifra no puede ser cero a menos que el valor sea 0.
	`void tciSetIntNumberOfDigits (TciValue inst, unsigned long int dig_num)`	Fija la cantidad de cifras de un valor entero.
	`void tciSetIntSign (TciValue inst, Boolean sign)`	Pone el signo + (true) o – (false).
	`void tciSetIntDigit (TciValue inst, unsigned long int position, char digit)`	Fija el valor de la cifra en 'position', donde la posición 0 indica la cifra menos significativa.
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)`	
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)`	

Interfaz IDL TCI	Representación ANSI-C	Notas y comentarios
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)`	
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)`	Devuelve un arreglo de los nombres de campos terminado por NULL.

Interfaz IDL TCI	Representación ANSI-C	Notas y comentarios
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)`	Devuelve un arreglo de los nombres de campos terminado por NULL.
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 Interfaz de registro

Interfaz IDL TCI	Representación ANSI-C	Notas y comentarios
Plantilla TciValue		
TBoolean isOmit()	`Boolean tciIsOmit(TciValueTemplate inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
TBoolean isAny()	`Boolean tciIsAny(TciValueTemplate inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
TBoolean isAnyOrOmit()	`Boolean tciIsAnyOrOmit(TciValueTemplate inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
TString getTemplateDef()	`String tciGetTemplateDef(TciValueTemplate inst)`	Tipo cadena reutilizado del IDL (recomendación del OMG)
Plantilla TciNonValue		
TBoolean isAny()	`Boolean tciIsAny(TciNonValueTemplate inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
TBoolean isAll()	`Boolean tciIsAll(TciNonValueTemplate inst)`	Tipo booleano reutilizado del IDL (recomendación del OMG)
TString getTemplateDef()	`String tciGetTemplateDef(TciNonValueTemplate inst)`	Tipo cadena reutilizado del IDL (recomendación del OMG)

9.4 Interfaces de operación

Requerida por 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()`	
Proporcionada por 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)`	
Requerida por 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)`	
Proporcionada por TCI-CD		

Value decode
(in TriMessageType message,
in Type decodingHypothesis)
	`TciValue tciDecode`
	`(BinaryString message,`
	`TciType decHypothesis)`
		Tipo BinaryString reutilizado de TRI

TriMessageType encode (in Value value)	`BinaryString tciEncode(TciValue value)`	
Requerida por 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)	
Proporcionada por 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 TriFirmaIdType signature, in TciParameterListType parameterList, in Value returnValue)	void tciReplyConnectedMC (TriPortId sender, TriComponentIdList receivers, TriFirmaIdType 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)	
Proporcionada por 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 TriFirmaIdType 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, TriFirmaId 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 line, in TriComponentIdType c, in TriMessageType msg, in TriStatusType decoderFailure, in Value val, in TString codec)	void tliDecode (String am, int ts, String src, int line, 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 Información

TCI IDL ADT	Representación ANSI-C (definición de tipo)	Notas y comentarios
TciModuleIdType	**QualifiedName**	
TciModuleParameterType	typedef struct TciModuleParameterType { String parName; TciValue defaultValue; } TciModuleParameterType;	
TciModuleParameterListType	typedef struct TciModuleParameterListType { long int length; TciModuleParameterType *modParList; } TciModuleParameterListType;	
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;	Se debe interpretar una longitud 0 como "lista vacía".
TciParameterTypeListType	typedef struct TciParameterTypeListType { long int length; TciType *parList; } TciParameterTypeListType;	Se debe interpretar una longitud 0 como "lista vacía".
TciTestCaseIdListType	typedef struct TciTestCaseIdListType { long int length; QualifiedName *idList; } TciTestCaseIdListType;	Se debe interpretar una longitud 0 como "lista vacía".

TCI IDL ADT	Representación ANSI-C (definición de tipo)	Notas y comentarios
TciModuleIdType	**QualifiedName**	

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;
		Se debe interpretar una longitud 0 como "lista vacía".

9.6 Varios

Concepto TCI	Representación ANSI-C	Notas y comentarios
Representación Verdict		
NONE	`const int TCI_VERDICT_NONE = 0`	Puesto que la interfaz TciVerdictValue se define en términos de enteros, se ha de alcanzar un acuerdo acerca de cuál valor define cuál veredicto.
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`	
Representación Objid		
Objid	`typedef struct TciObjidValue { long int length; TciObjidElem *elements; } TciObjidValue;`	Puesto que el valor Objid se devuelve "tal como es" a través de la interfaz de valor Objid, se debe definir una representación.
TciObjidElem	`typedef struct TciObjidElemValue { char* elem_as_ascii; long int elem_as_number; void* aux; } TciObjidElemValue;`	
Representación CharstringValue		
TciCharString	`typedef struct TciCharStringValue { unsigned long int length; char* string; } TciCharStringValue`	
Representación Universal Character[string]		
Universal Char	`typedef unsigned char[4] TciUCValue`	
Universal Charstring	`typedef struct TciUCStringValue { unsigned long int length; TciUCType *string; } TciUCStringValue;`	

10 Correspondencia XML W3C

10.1 Introducción

En esta cláusula se describe la correspondencia XML TCI [5], [6] y [7] para la interfaz de registro de la TCI. En dicha correspondencia XML se indica cómo se hacen corresponder con XML las definiciones IDL de la cláusula 7. En el anexo B se presentan las definiciones de esquema para esta correspondencia.

10.2 Alcances

Se hace corresponder el módulo IDL **`tciInterface`** con un esquema XML cuyo espacio de nombre es `http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI`.

Este esquema, a su vez, utiliza otros esquemas:

- http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes para la correspondencia de tipos simples con XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Types para la correspondencia de tipos estructurados con XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Values para la correspondencia de valores con XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates para la correspondencia de plantillas con XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Events para la correspondencia de eventos de registro con XML.

10.3 Correspondencia de tipo

10.3.1 Correspondencia de tipos simples

10.3.1.1 TBoolean

El tipo IDL **`TBoolean`** se hace corresponder con el tipo básico xsd `boolean`.

10.3.1.2 TString

El tipo IDL **`TString`** se hace corresponder con el tipo básico xsd `string`.

10.3.1.3 TInteger

El tipo IDL **`TInteger`** se hace corresponder con el tipo básico xsd `integer`.

10.3.1.4 TriTimerDurationType

El tipo **`TriTimerDurationType`** se hace corresponder con el tipo básico xsd `float`.

10.3.1.5 TciParameterPassingModeType

El tipo IDL **`TciParameterPassingModeType`** se hace corresponder con el tipo básico xsd `string` con valores de enumeración `'in'`, `'out'` e `'inout'`.

10.3.1.6 TriStatusType

El tipo **`TriStatusType`** se hace corresponder con el tipo básico xsd `string` con valores de enumeración `'TRI_Ok'` y `'TRI_Error'`.

10.3.1.7 TciStatusType

El tipo **`TriStatusType`** se hace corresponder con el tipo básico xsd `string` con valores de enumeración `'TCI_Ok'` y `'TCI_Error'`.

10.3.2 Correspondencia de tipo complejo

10.3.2.1 TriPortIdType

`TriPortIdType` corresponde al siguiente tipo complejo:

```
<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>
```

Elementos:

- `comp` El identificador de componente TRI;
- `port` La identificación del puerto.

Atributos:

- ninguno

10.3.2.2 TriComponentIdType

`TriComponentIdType` corresponde al siguiente tipo complejo:

```
<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>
```

Elementos:

- `id` El identificador de componente TRI;
- `null` El identificador `null`. Se debe emplear si no hay identificador de componente TRI.

Atributos:

- ninguno.

10.3.2.3 TriComponentIdListType

`TriComponentIdListType` corresponde al siguiente tipo complejo:

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

Elementos:

- `comp` Los identificadores de componentes TRI en dicha lista.

Atributos:

- ninguno.

10.3.2.4 Port

`Port` corresponde al siguiente tipo complejo:

```
<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>
```

Elementos:

- `id` El identificador de puerto;
- `port` El índice de puerto.

Atributos:

- ninguno.

10.3.2.5 Id

`Id` se emplea como identificador de componentes, puertos y temporizadores y se hace corresponder al siguiente tipo complejo:

```
<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>
```

Elementos:

- `name` El nombre del componente, puerto o temporizador;
- `id` La representación interna del componente, puerto o temporizador;
- `type` El tipo de componente, puerto o temporizador.

Atributos:

- ninguno.

10.3.2.6 TriMessageType

`TriMessageType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` El mensaje codificado.

Atributos:

- ninguno.

10.3.2.7 TriParameterType

`TriParameterType` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Elementos:

- `val` El parámetro codificado.

Atributos:

- `name` El nombre de parámetro;
- `mode` El modo de paso de parámetro.

10.3.2.8 TriParameterListType

`TriParameterListType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `par` Los parámetros en esta lista.

Atributos:

- ninguno.

10.3.2.9 TriAddressType

`TriAddressType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` El valor de dirección.

Atributos:

- ninguno.

10.3.2.10 TriAddressListType

`TriAddressListType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `addr` Las direcciones en esta lista.

Atributos:

- ninguno.

10.3.2.11 TriExceptionType

`TriExceptionType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` La excepción.

Atributos:

- ninguno.

10.3.2.12 TriSignatureIdType

`TriSignatureIdType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` La firma.

Atributos:

- ninguno.

10.3.2.13 TriAddressType

`TriAddressType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` La dirección dentro del SUT.

Atributos:

- ninguno.

10.3.2.14 TriTimerIdType

`TriTimerIdType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `id` La identificación del temporizador.

Atributos:

- ninguno.

10.3.2.15 TriTimerDurationType

`TriTimerDurationType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` La duración del temporizador.

Atributos:

- ninguno.

10.3.2.16 QualifiedName

`QualifiedName` se emplea con parámetros, variables de módulo, etc. completamente calificados, y se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `moduleName` El nombre de módulo del módulo TTCN-3.
- `baseName` El nombre del objeto que está completamente calificado.

Atributos:

- ninguno.

10.3.2.17 TciBehaviourIdType

`TciBehaviourIdType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `name` El nombre calificado del comportamiento.

Atributos:

- ninguno.

10.3.2.18 TciTestCaseIdType

`TciTestCaseIdType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `name` El nombre calificado del caso de prueba.

Atributos:

- ninguno.

10.3.2.19 TciParameterType

`TciParameterType` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Elementos:

- `val` El parámetro codificado.

Atributos:

- `name` El nombre de parámetro.
- `mode` El modo de paso de parámetro.

10.3.2.20 TciParameterListType

`TciParameterListType` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `par` Los parámetros en esta lista.

Atributos:

- ninguno.

10.3.3 Correspondencia de valor abstracto

10.3.3.1 Value

`Value` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- `name` El nombre del valor, si es conocido.
- `type` El tipo del valor, si es conocido.
- `module` El módulo del valor, si es conocido.

10.3.3.2 IntegerValue

`IntegerValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor entero como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.3 FloatValue

`FloatValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor float como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.4 BooleanValue

`BooleanValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor booleano como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.5 ObjidValue

`ObjidValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor objid como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.6 VerdictValue

`VerdictValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor verdict como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.7 BitstringValue

`BitstringValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor bitstring como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.8 HexstringValue

`HexstringValue` se hace corresponder al siguiente tipo complejo:

```
<xsd:complexType name="HexstringValue">
    <xsd:simpleContent>
```

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

Contenido simple:

- `base` El valor hexstring como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.9 OctetstringValue

`OctetstringValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor octetstring como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.10 CharstringValue

`CharstringValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor charstring como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.11 UniversalCharstringValue

`UniversalCharstringValue` se hace corresponder al siguiente tipo complejo:

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

Contenido simple:

- `base` El valor charstring universal como cadena.
- `extension` Los mismos atributos que Value.

10.3.3.12 RecordValue

`RecordValue` se hace corresponder al siguiente tipo complejo:

```
    <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>
```

Secuencia de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.13 RecordOfValue

`RecordOfValue` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de la secuencia de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.

- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.14 SetValue

`SetValue` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Secuencia de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.15 SetOfValue

`SetOfValue` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de la secuencia de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.16 EnumeratedValue

`EnumeratedValue` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `element` El valor de enumeración.

Atributos:

- Los mismos atributos que Value.

10.3.3.17 UnionValue

`UnionValue` se hace corresponder al siguiente tipo complejo:

```
    <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>
```

Selección de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.18 AnytypeValue

`AnytypeValue` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.
- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `address` Un valor address.

Atributos:

- Los mismos atributos que Value.

10.3.3.19 AddressValue

`AddressValue` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de elementos:

- `integer` Un valor integer.
- `float` Un valor float.
- `boolean` Un valor boolean.
- `objid` Un valor objid.
- `verdicttype` Un valor verdicttype.
- `bitstring` Un valor bitstring.
- `hexstring` Un valor hexstring.

- `octetstring` Un valor octetstring.
- `charstring` Un valor charstring.
- `universal_charstring` Un valor charstring universal.
- `record` Un valor record.
- `record_of` Un valor record of.
- `set` Un valor set.
- `set_of` Un valor set of.
- `enumerated` Un valor enumerated.
- `union` Un valor union.
- `anytype` Un valor anytype.

Atributos:

- Los mismos atributos que Value.

10.3.4 Correspondencia de tipos de registro abstractos

Se definen otros tipos para facilitar el registro de correspondencias entre valores y plantillas.

10.3.4.1 TciValueTemplate

`TciValueTemplate` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de elementos:

- `integer` Una plantilla integer.
- `float` Una plantilla float.
- `boolean` Una plantilla boolean.
- `objid` Una plantilla objid.
- `verdicttype` Una plantilla verdicttype.
- `bitstring` Una plantilla bitstring.
- `hexstring` Una plantilla hexstring.
- `octetstring` Una plantilla octetstring.
- `charstring` Una plantilla charstring.

- `universal_charstring` Una plantilla universal charstring.
- `record` Una plantilla record.
- `record_of` Una plantilla record of.
- `set` Una plantilla set.
- `set_of` Una plantilla set of.
- `enumerated` Una plantilla enumerated.
- `union` Una plantilla union.
- `anytype` Una plantilla anytype.
- `address` Una plantilla address.
- `omit` Una plantilla omit.
- `any` Una plantilla any.
- `anyoromit` Una plantilla anyoromit.
- `templateDef` Una definición de plantilla complex.

Atributos:

- ninguno.

10.3.4.2 TciNonValueTemplate

`TciNonValueTemplate` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Selección de elementos:

- `any` Una plantilla any.
- `all` Una plantilla all.
- `templateDef` Una definición de plantilla complex.

Atributos:

- ninguno

10.3.4.3 TciValueList

`TciValueList` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `val` Los valores en la lista Value.

Atributos:

- ninguno.

10.3.4.4 TciValueDifference

`TciValueDifference` se hace corresponder al siguiente tipo complejo:

```
<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>
```

Elementos:

- `desc` El motivo de la falta de correspondencia.
- `val` Una referencia al valor que no corresponde.
- `tmpl` Una referencia a la plantilla.

Atributos:

- ninguno.

10.3.4.5 TciValueDifferenceList

`TciValueDifferenceList` se hace corresponder al siguiente tipo complejo:

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

Elementos:

- `diff` Las diferencias de valor/plantilla en la lista de diferencias de valor.

Atributos:

- ninguno.

10.4 Correspondencia de operaciones en la interfaz de registro

Toda operación proporcionada en la interfaz de registro tiene una definición de tipo complejo correspondiente en XML. Dichas definiciones de tipo complejo son extensiones de Event.

10.4.1 Event

`Event` se hace corresponder al siguiente tipo complejo:

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

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

Elementos:

- `am` Un mensaje, que se debe utilizar para suministrar más información en el registro.

Atributos:

- `ts` La hora en que se produce el evento.
- `src` El archivo origen de la especificación de prueba.
- `line` El número de la línea en que se efectúa la petición.
- `name` El nombre del componente que produce este evento.
- `id` El id del componente que produce este evento.
- `type` El tipo del componente que produce este evento.

10.4.2 Correspondencia de operaciones

A continuación se suministra la correspondencia de operaciones.

Proporcionada por 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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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>

Proporcionada por TCI-TL	
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>
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>

Proporcionada por TCI-TL	
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>

Proporcionada por 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 Casos de utilización

En esta cláusula se presentan ejemplos que deberían ser útiles a usuarios de la TCI y a fabricantes de equipos, siempre y cuando la TCI acepte las semánticas de las operaciones definidas en esta Recomendación.

Estos casos se definen en términos de diagramas de secuencia UML, que muestran las interacciones entre entidades TCI. Se da una explicación de cada caso y, cuando procede, se añade un fragmento TTCN-3 correspondiente.

11.1 Inicialización, recolección de información y registro

11.1.1 Caso de utilización: inicialización

En el caso de la figura 9 se presenta la fase de inicialización de un sistema de pruebas en el que se ha de escoger un módulo TTCN-3 para ejecución. Para comenzar, se debe establecer un módulo raíz, utilizando `tciRootModule`, cuyos parámetros se pueden obtener a través de `tciGetModuleParameters`. Se puede utilizar la información de parámetro de módulo para solicitar al usuario del sistema de pruebas valores concretos para cada parámetro de módulo. La lista de casos de prueba disponibles en el módulo raíz se obtiene mediante `tciGetTestCases`. Los casos de prueba se pueden

ejecutar directamente desde la gestión de pruebas. Para obtener sus parámetros y su interfaz de sistema de pruebas se emplean, respectivamente, las operaciones `tciGetTestCaseParameters` y `tciGetTestCaseTSI`.

11.1.1.1 Diagrama de secuencia

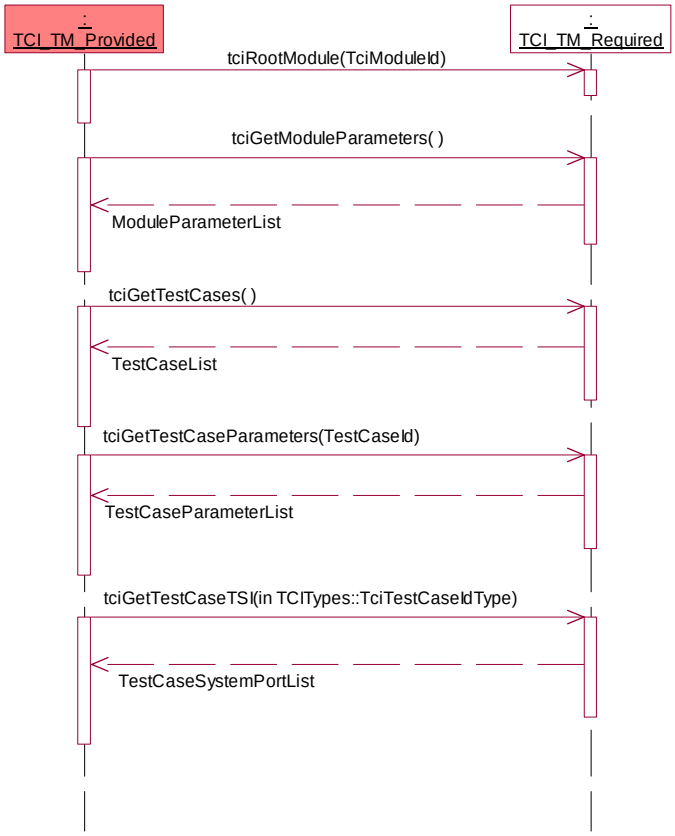

Figura 9/Z.145 – Escenario de utilización – Inicialización

11.1.1.2 Fragmento TTCN-3

La inicialización está fuera del alcance de TTCN-3.

11.1.2 Escenario de utilización: solicitud de parámetros de módulo

En la figura 10 se muestra cómo un componente de pruebas solicita el valor real de un parámetro de módulo necesario para la ejecución de su comportamiento de prueba. Se solicita primero el tipo del parámetro de módulo y, luego, el TM puede construir el valor y pasarlo al TE.

11.1.2.1 Diagrama de secuencia

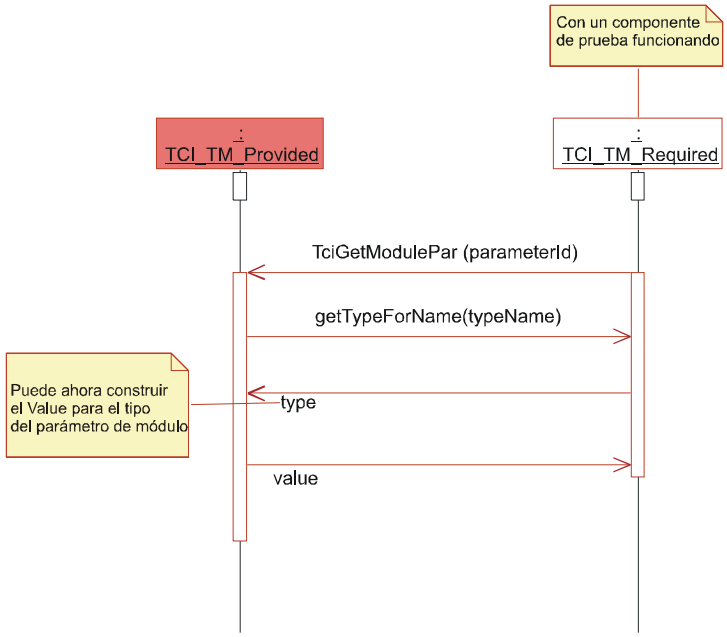

Figura 10/Z.145 – Escenario de utilización – Solicitud de parámetros de módulo

11.1.2.2 Fragmento TTCN-3

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

11.1.3 Escenario de utilización: registro

En la figura 11 se describe el registro de información durante la ejecución de un comportamiento de prueba por un componente de prueba. El mensaje que se ha de registrar se propaga al registro de pruebas.

11.1.3.1 Diagrama de secuencia

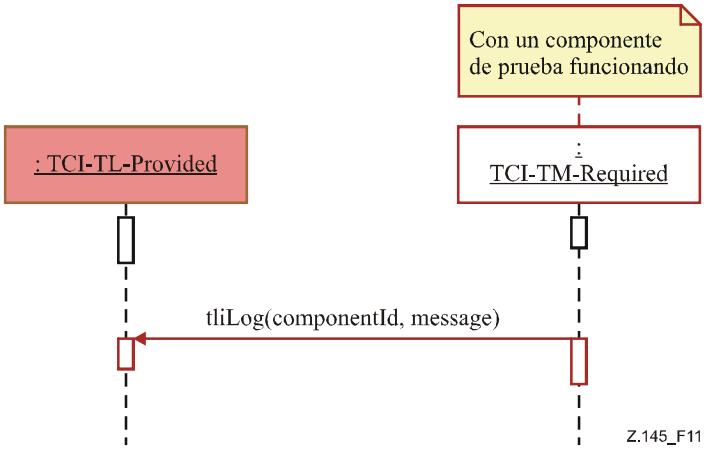

Figura 11/Z.145 – Escenario de utilización – Registro

11.1.3.2 Fragmento TTCN-3

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

11.2 Ejecución de control y casos de pruebas

11.2.1 Escenario de utilización: ejecución de control

En la figura 12 se muestra la secuencia de operaciones necesarias para la ejecución de la parte de control de un módulo TTCN-3. Se escoge primero el módulo que contiene la parte de control, luego se lo inicia y ejecuta hasta que el TE termina la ejecución.

11.2.1.1 Diagrama de secuencia

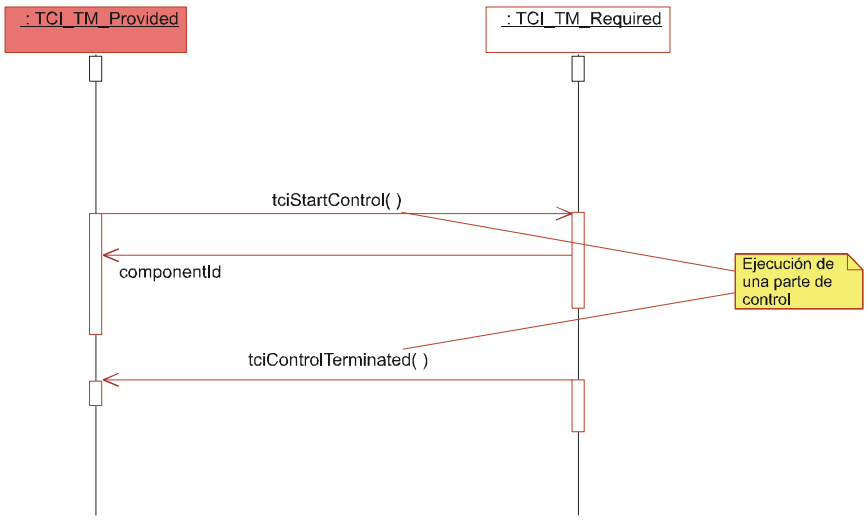

Figura 12/Z.145 – Escenario de utilización – Ejecución de control

11.2.1.2 Fragmento TTCN-3

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

11.2.2 Escenario de utilización: ejecución de caso de prueba en la parte de control

En la figura 13 se muestra cómo se ejecuta un caso de prueba en la parte de control.

11.2.2.1 Diagrama de secuencia

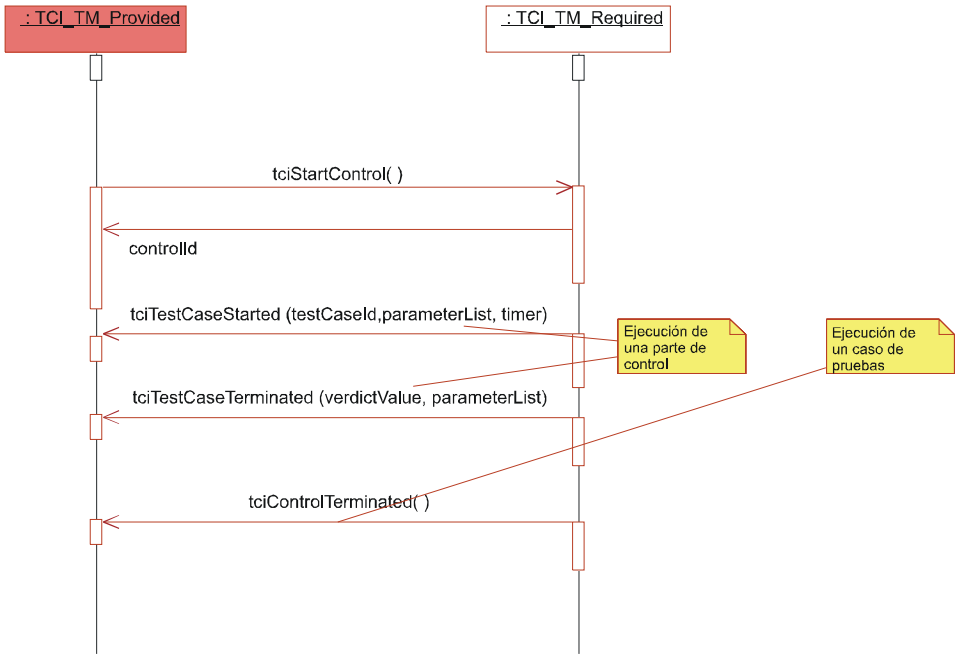

Figura 13/Z.145 – Escenario de utilización – Ejecución de caso de prueba en la parte de control

11.2.2.2 Fragmento TTCN-3

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

11.2.3 Escenario de utilización: ejecución directa de caso de prueba

En la figura 14 se muestra cómo ejecutar directamente un caso de prueba desde la gestión de prueba fuera de la parte de control. Tras haber escogido el módulo TTCN-3, que contiene el caso de prueba que se ha de ejecutar, se solicita el inicio de éste. Cuando se completa la ejecución, el TE informa de la terminación a la gestión de prueba.

11.2.3.1 Diagrama de secuencia

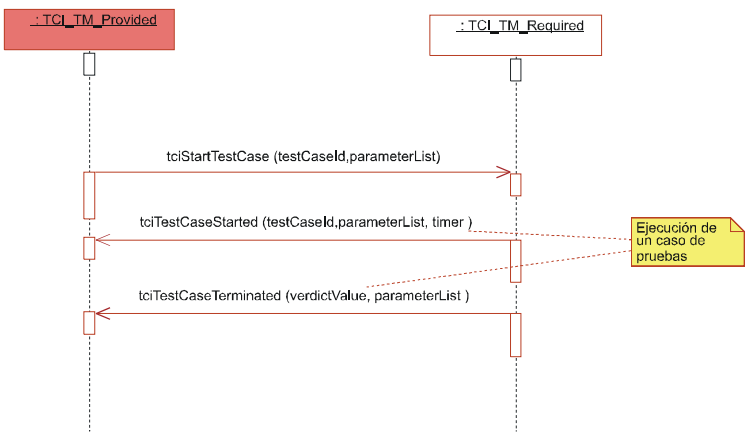

Figura 14/Z.145 – Escenario de utilización – Ejecución directa de caso de prueba

11.2.3.2 Fragmento TTCN-3

La ejecución directa de casos de prueba está fuera del alcance de TTCN-3.

11.2.4 Escenario de utilización: información a la TRI de la ejecución del caso de prueba

En la figura 15 se muestra cómo se informa la TRI de la ejecución de un caso de prueba, de tal manera que pueda, si fuere necesario, configurar e inicializar los puertos de sistema requeridos. Se debe generar la solicitud de ejecución de caso de prueba antes de que se inicie el comportamiento de prueba del MTC del caso de prueba en cuestión.

11.2.4.1 Diagrama de secuencia

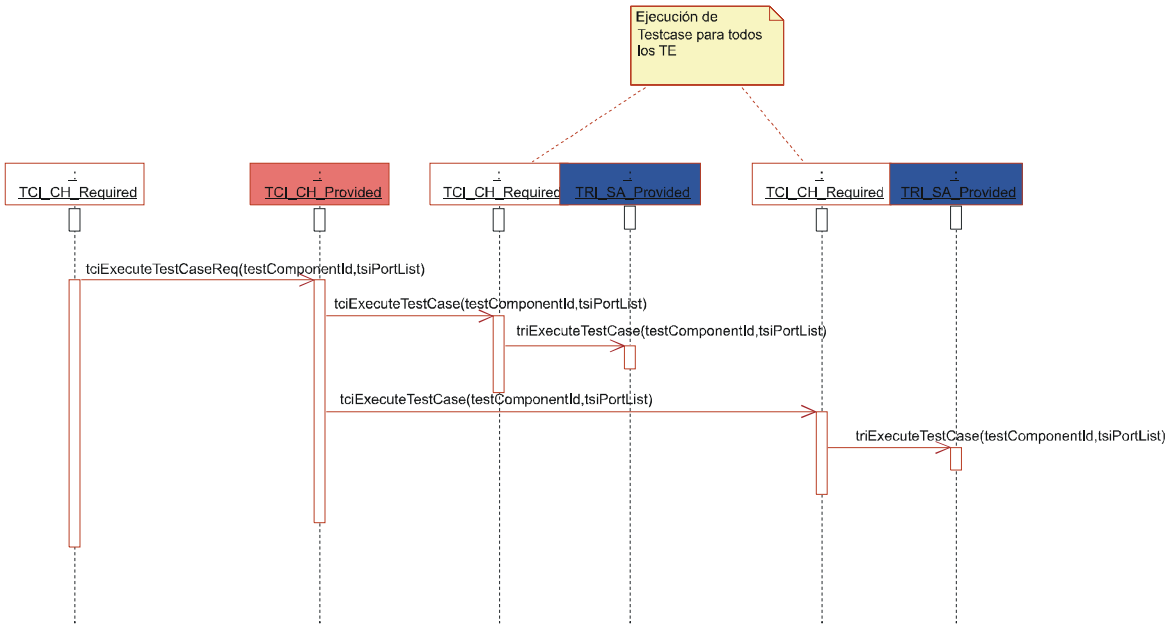

Figura 15/Z.145 – Escenario de utilización – Información a la TRI de la ejecución del caso de prueba

11.2.4.2 Fragmento TTCN-3

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

11.3 Procesamiento de componentes

11.3.1 Escenario de utilización: creación local de componente de control

En la figura 16 se describe la creación del componente de control en el mismo nodo en que se encuentra la interfaz de usuario con la gestión de pruebas TCI-TM. Siempre que se ejecuta la parte de control de un módulo TTCN-3, se crea un componente de control. Siempre que la gestión de pruebas TCI-TM inicia la parte de control, se envía una solicitud de creación de componente al TCI-CH, que la hace pasar al TE, donde debe crearse el componente de control. En este caso, se trata del TE en el mismo nodo. Se devuelve el identificador del componente de control y se entrega al TCI-TM. Luego, se emplea dicho identificador para iniciar el comportamiento de la parte de control en el componente de control.

11.3.1.1 Diagrama de secuencia

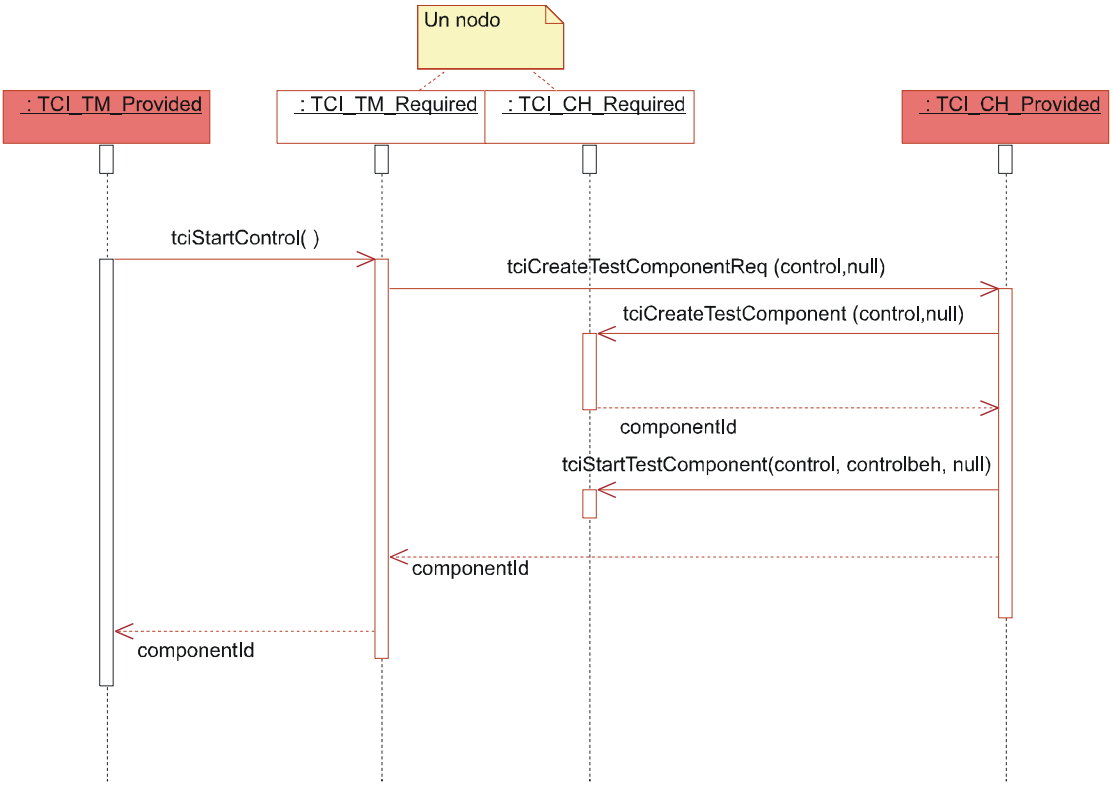

Figura 16/Z.145 – Escenario de utilización – Creación local de componente de control

11.3.1.2 Fragmento TTCN-3

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

11.3.2 Escenario de utilización: creación distante de componente de control

En la figura 17 se describe la creación del componente de control en un nodo diferente de aquel en que se encuentra la interfaz de usuario con la gestión de pruebas TCI-TM. Siempre que se ejecuta la parte de control de un módulo TTCN-3, se crea un componente de control. Siempre que la gestión de pruebas TCI-TM inicia la parte de control, se envía una solicitud de creación de componente al TCI-CH, que la hace pasar al TE, donde debe crearse el componente de control. En este caso, se trata del TE en un nodo distante. Se devuelve el identificador del componente de control y se entrega al TCI-TM. Luego, se emplea dicho identificador para iniciar el comportamiento de la parte de control en el componente de control.

11.3.2.1 Diagrama de secuencia

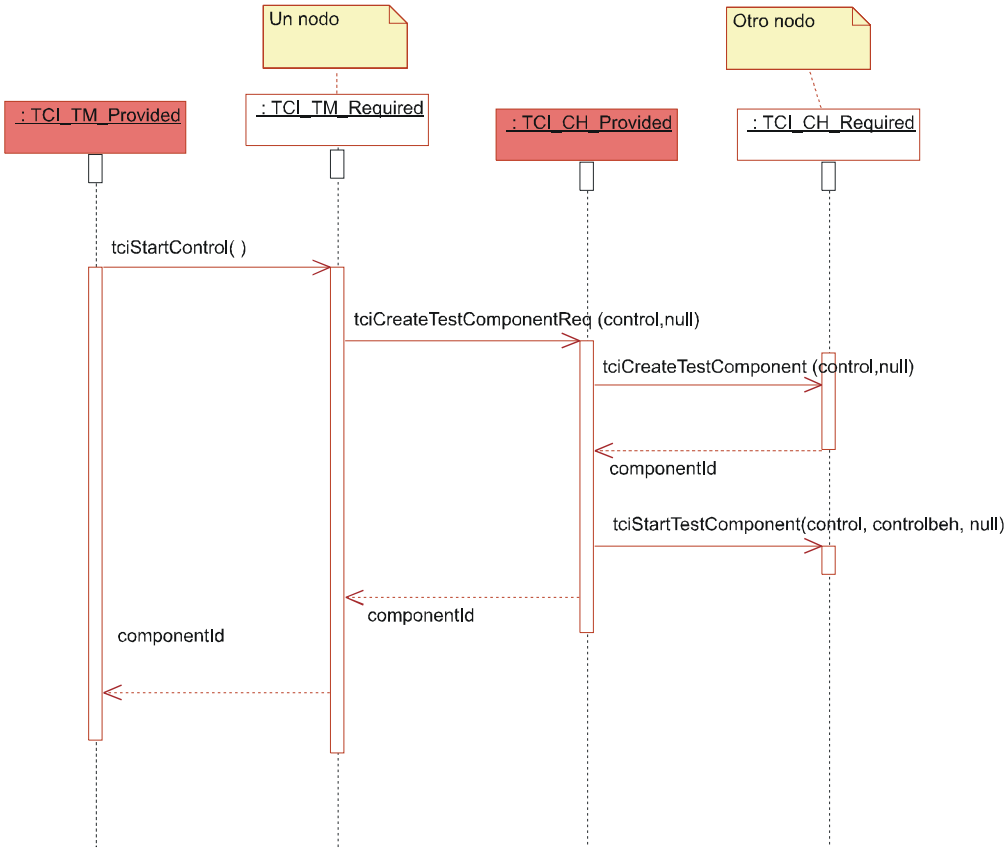

Figura 17/Z.145 – Escenario de utilización – Creación distante de componente de control

11.3.2.2 Fragmento TTCN-3

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

11.3.3 Escenario de utilización: creación local del MTC

En la figura 18 se describe la creación local del componente principal de prueba. La expresión "local" tiene en este caso dos significados, a saber:

1) en el mismo nodo en que se encuentra la interfaz de usuario con la gestión de pruebas TCI-TM (cuando el caso de pruebas se inicia directamente); o
2) en el mismo nodo en que se encuentra el componente de control (cuando el caso de pruebas se ejecuta desde una parte de control).

Siempre que se ejecuta un caso de prueba, se crea un componente principal de prueba: se envía una solicitud de creación al TCI-CH, que la hace pasar al TE, donde tiene lugar dicha creación. En este caso, se trata del TE en el mismo nodo. Se devuelve el identificador del componente principal de prueba y se entrega al TCI-TM. Luego, se emplea dicho identificador para iniciar el comportamiento de caso de prueba en el componente principal de prueba (si bien esto no se muestra en la figura, se trata de la misma manera que los escenarios de 11.3.5 y 11.3.6).

11.3.3.1 Diagrama de secuencia

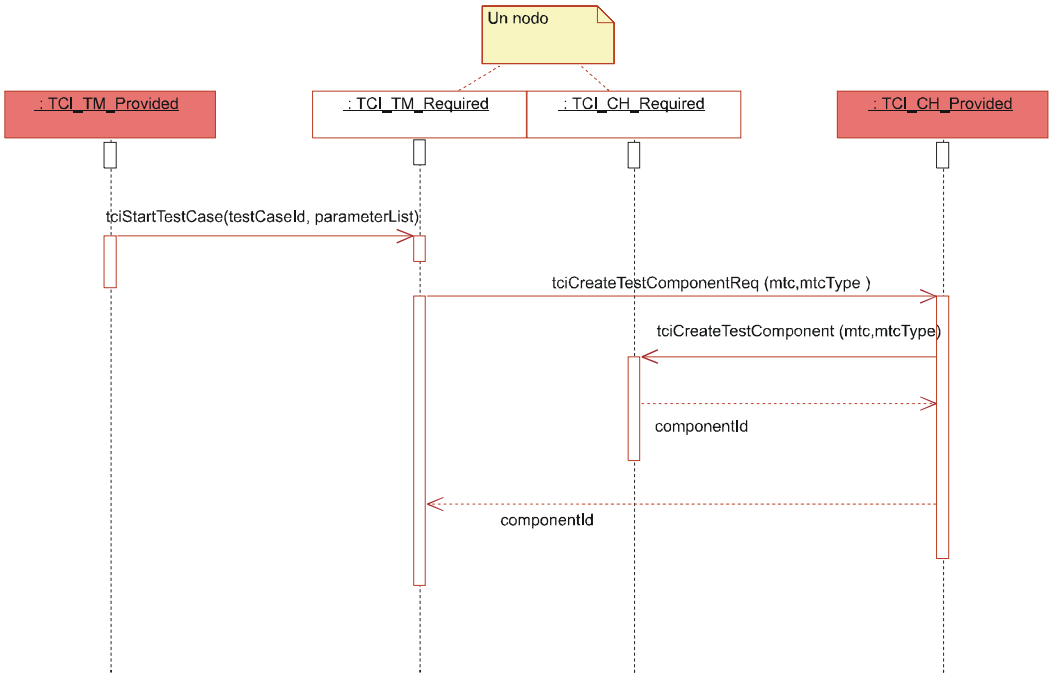

Figura 18/Z.145 – Escenario de utilización – Creación local del MTC

11.3.3.2 Fragmento TTCN-3

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

11.3.4 Escenario de utilización: creación distante del MTC

En la figura 19 se describe la creación distante del componente principal de prueba. La expresión "distante" tiene en este caso dos significados, a saber:

1) en un nodo diferente de aquel en que se encuentra la interfaz de usuario con la gestión de prueba TCI-TM (cuando el caso de pruebas se inicia directamente); o
2) en el mismo nodo en que se encuentra el componente de control (cuando el caso de pruebas se ejecuta desde una parte de control).

Siempre que se ejecuta un caso de prueba, se crea un componente principal de prueba: se envía una solicitud de creación al TCI-CH, que la hace pasar al TE, donde tiene lugar dicha creación. En este caso, se trata del TE en otro nodo. Se devuelve el identificador del componente principal de prueba y se entrega al TCI-TM. Luego, se emplea dicho identificador para iniciar el comportamiento de caso de prueba en el componente principal de prueba (si bien esto no se muestra en la figura, se trata de la misma manera que los escenarios de 11.3.5 y 11.3.6).

11.3.4.1 Diagrama de secuencia

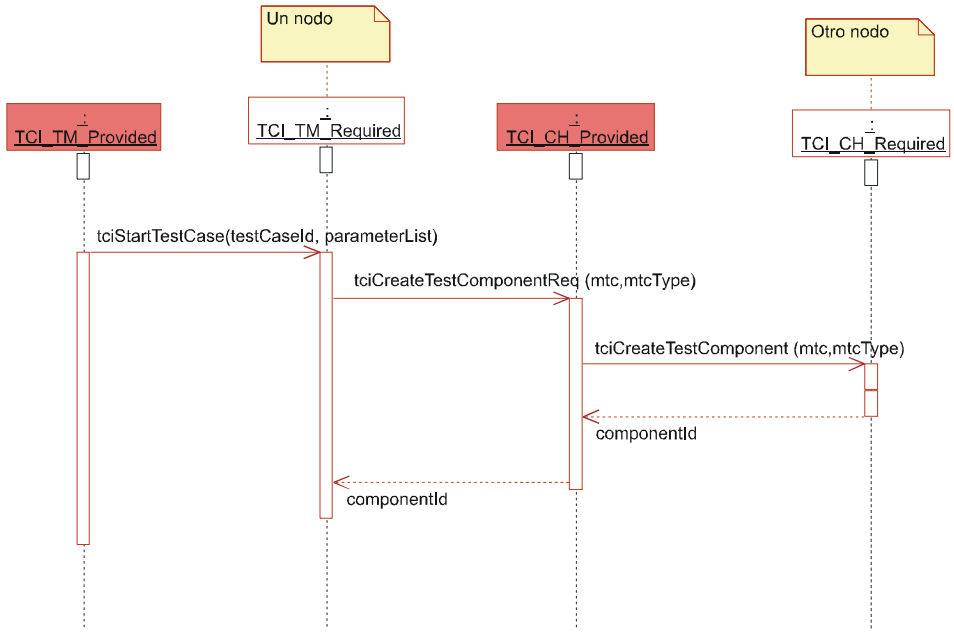

Figura 19/Z.145 – Escenario de utilización – Creación distante del MTC

11.3.4.2 Fragmento TTCN-3

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

11.3.5 Escenario de utilización: procesamiento de componente cuando se ejecuta el caso de prueba dentro de la parte control

En la figura 20 se describe el procesamiento de componentes para la ejecución de un caso de prueba dentro de la parte de control. Cuando se inicia la parte de control, se crea un componente de control cuyo identificador se devuelve a la gestión de prueba. Para cada caso de prueba que se ha de ejecutar dentro de la parte de control, se crea un componente principal de prueba y el identificador de componente se devuelve al componente de control. Luego, se inicia el comportamiento de caso de prueba en el componente principal de prueba y se informa a la gestión de prueba acerca del comienzo del caso de prueba. Cuando termina el componente principal de prueba, se propaga una petición de evaluación de la terminación del componente principal de prueba, junto con el veredicto local de componente principal de prueba, para facilitar el cálculo del veredicto global de componente principal de prueba y con el fin de que se pueda informar sobre la terminación de dicho caso.

11.3.5.1 Diagrama de secuencia

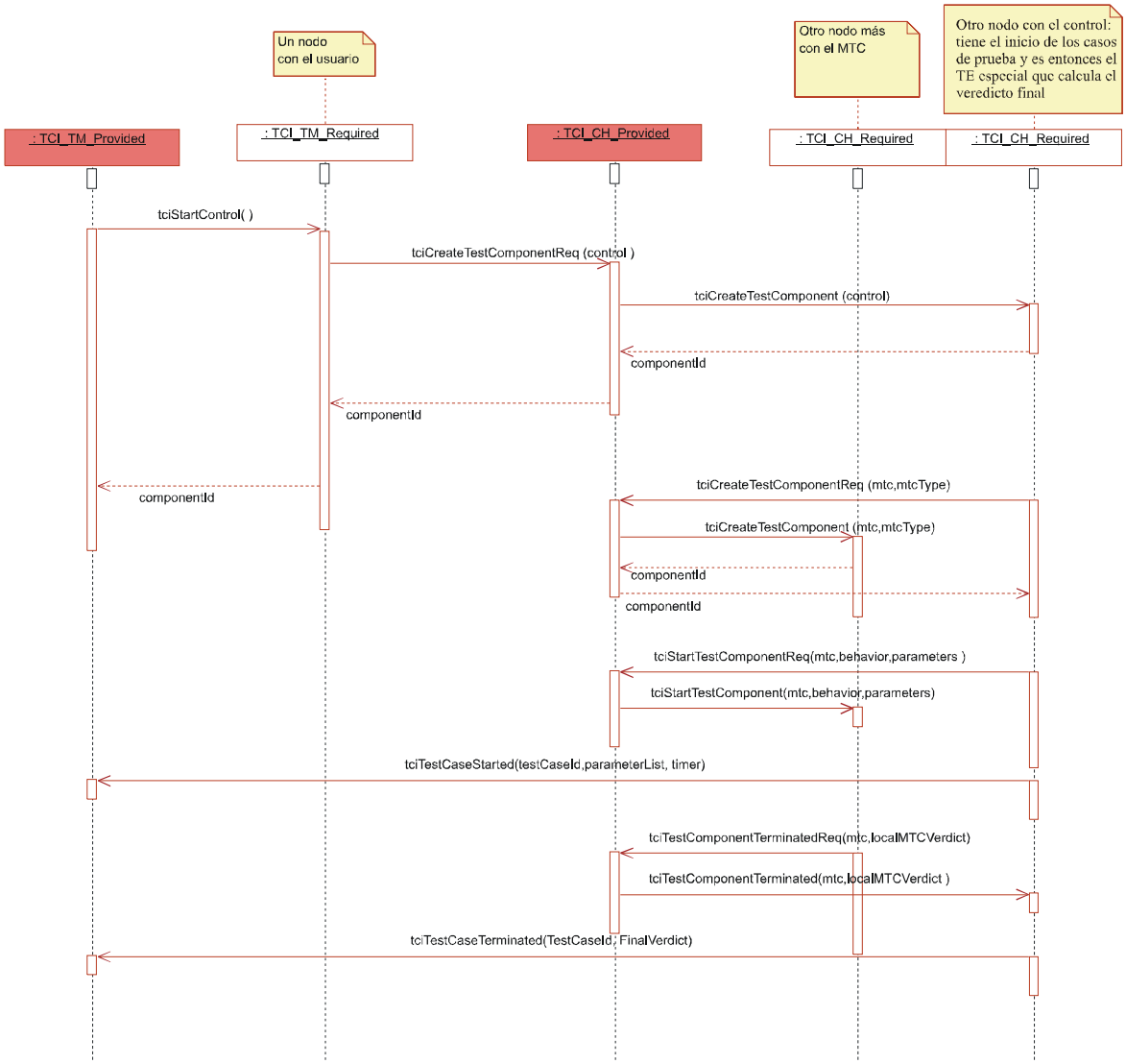

Figura 20/Z.145 – Escenario de utilización – Procesamiento de componente cuando se ejecuta el caso de prueba dentro de la parte control

11.3.5.2 Fragmento TTCN-3

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

11.3.6 Escenario de utilización: procesamiento de componente para la ejecución directa de un caso de prueba

En la figura 21 se indica cómo se procesan los componentes de prueba cuando se ejecuta directamente un caso de prueba, es decir fuera de una parte de control. Cuando se inicia un caso de prueba, se crea el componente principal de prueba y en él se inicia, en primer lugar, el comportamiento de prueba. Siempre que se utiliza un componente paralelo de prueba en un caso de prueba, éste se procesa de la misma manera: se inicia primero el componente paralelo, transmitiendo una petición de creación de componente de prueba a la entidad TCI-CH, que la pasa al TE donde se debe

crear dicho componente paralelo. Se devuelve el identificador del componente paralelo de prueba creado, el cual sirve para empezar el comportamiento PTC de la operación inicio. Cuando el PTC termina su ejecución, se emite, para informar al TCI-CH de dicha terminación, una petición de terminación de componente de prueba, junto con el veredicto local de prueba. Algo idéntico ocurre cuando termina el componente principal de prueba. Además, la terminación del componente principal de prueba conduce a la terminación global del caso de prueba.

11.3.6.1 Diagrama de secuencia

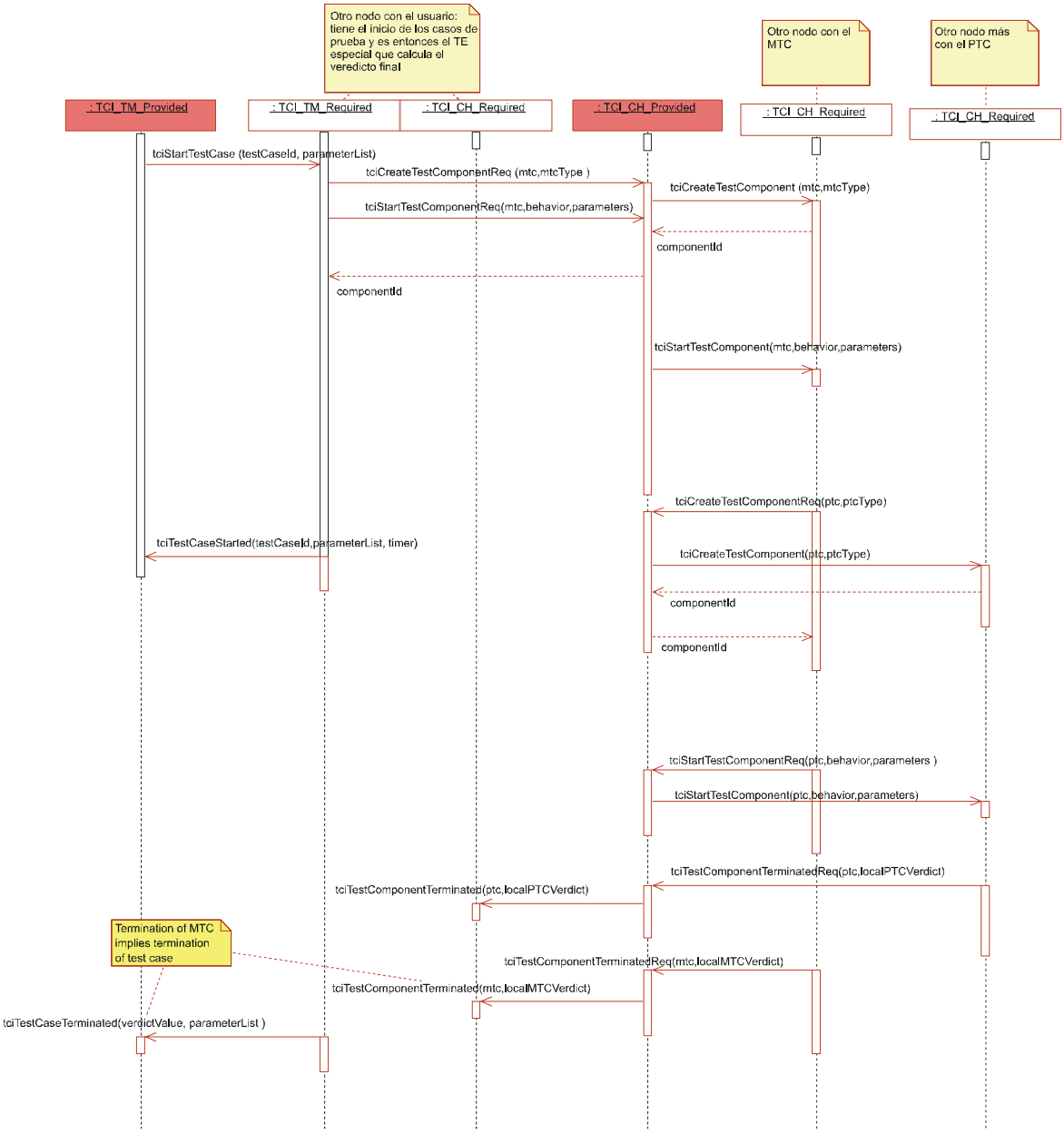

Figura 21/Z.145 – Escenario de utilización – Procesamiento de componente para la ejecución directa de un caso de prueba

11.3.6.2 Fragmento TTCN-3

```
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 Escenario de utilización: propagación de map/connect

En la figura 22 se indica cómo se hacen corresponder los puertos. Se propaga hacia el TE la solicitud de correspondencia de puerto, donde finalmente se ejecuta dicha correspondencia. La propagación de solicitudes de conexión funciona de una manera análoga.

11.3.7.1 Diagrama de secuencia

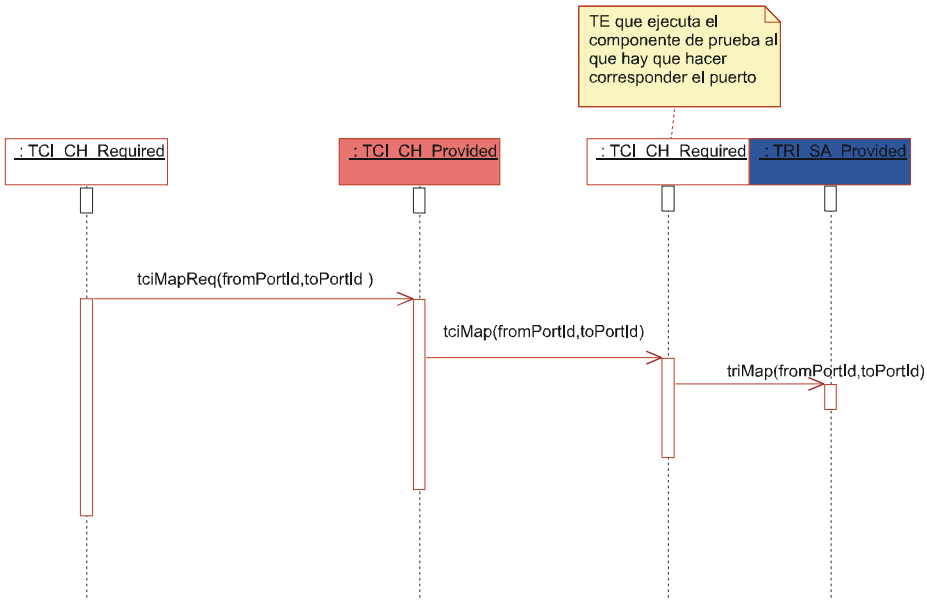

Figura 22/Z.145 – Escenario de utilización – propagación de map

11.3.7.2 Fragmento TTCN-3

```
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 Escenario de utilización: propagación de unmap/disconnect

En la figura 23 se indica cómo se terminan las correspondencias de puertos. Se propaga hacia el TE la solicitud de anulación de correspondencia de puerto, donde finalmente se ejecuta dicha operación. La propagación de solicitudes de desconexión funciona de una manera análoga.

11.3.8.1 Diagrama de secuencia

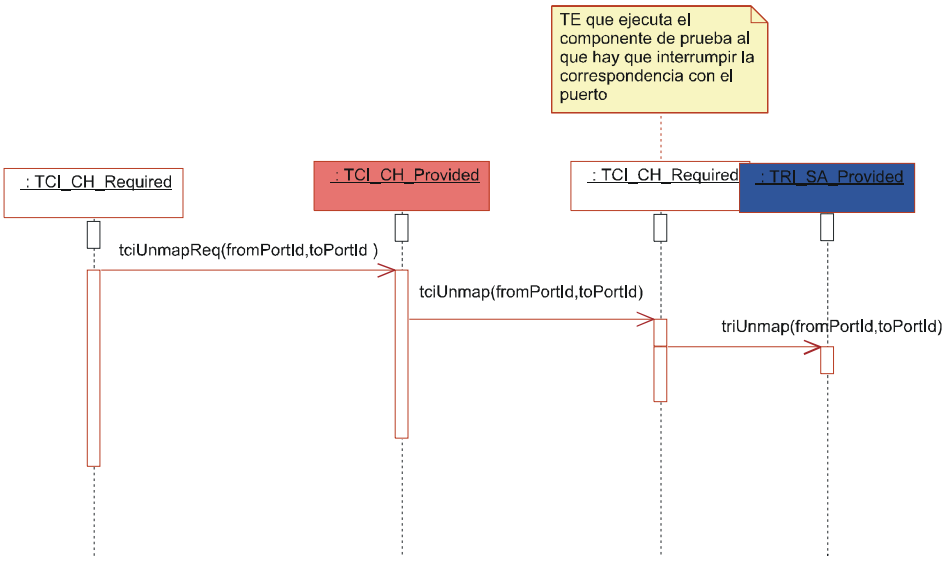

Figura 23/Z.145 – Escenario de utilización – Propagación de unmap

11.3.8.2 Fragmento TTCN-3

```
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 Terminación de casos de prueba y control

11.4.1 Escenario de utilización: interrupción de un caso de prueba

En la figura 24 se indica cómo la gestión de pruebas detiene un caso de prueba durante su ejecución. Una vez el TM ha recibido información sobre un caso de prueba iniciado, se puede solicitar una interrupción (stop) de dicho caso hasta tanto no se reciba información de que ha terminado. Tras la interrupción de un caso de prueba, se interrumpen todos los componentes paralelos de prueba y se reinicializa el sistema de pruebas.

11.4.1.1 Diagrama de secuencia

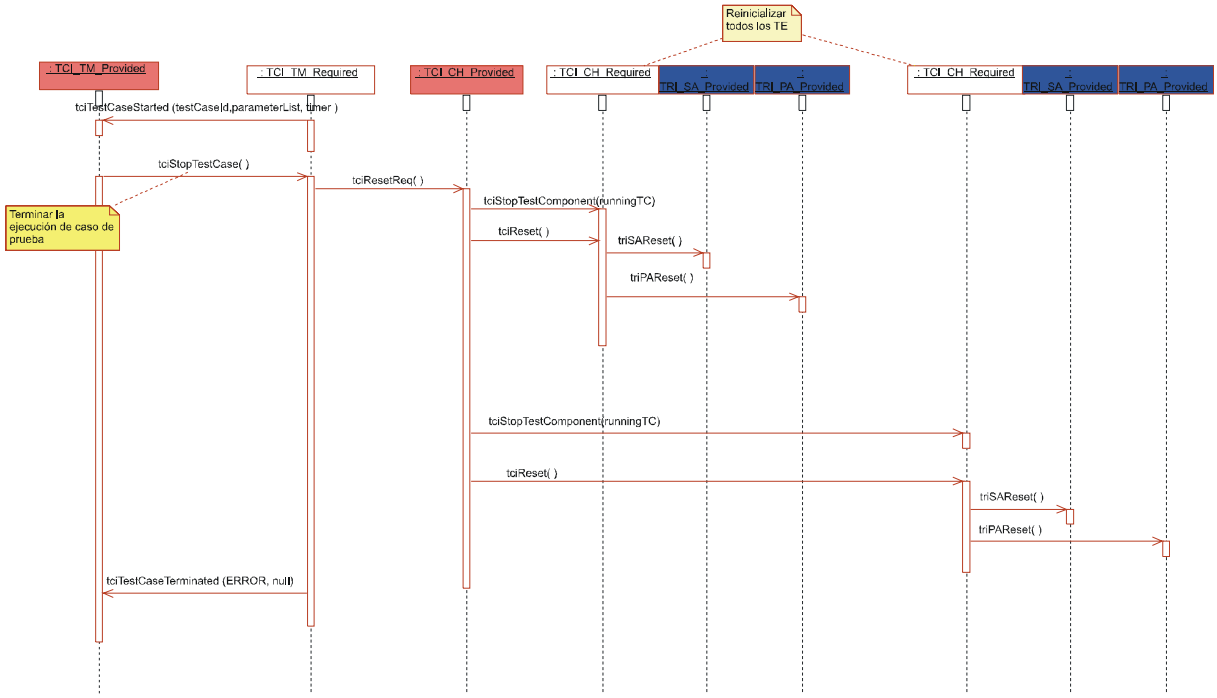

Figura 24/Z.145 – Escenario de utilización – Interrupción de un caso de prueba

11.4.1.2 Fragmento TTCN-3

No hay código TTCN-3 relacionado con la forma como el TM decide implementar una terminación de caso de prueba. Esto está fuera del alcance de TTCN-3.

11.4.2 Escenario de utilización: interrupción de control

En la figura 25 se indica cómo la gestión de pruebas detiene la parte de control durante su ejecución. Se puede detener una parte de control en cualquier momento entre su inicio y su terminación. Si la parte de control recibe una petición de interrupción de caso de prueba durante la ejecución de este último, el caso ha de interrumpirse. Además, se reinicializará el sistema de prueba, como se indica en la figura 24.

11.4.2.1 Diagrama de secuencia

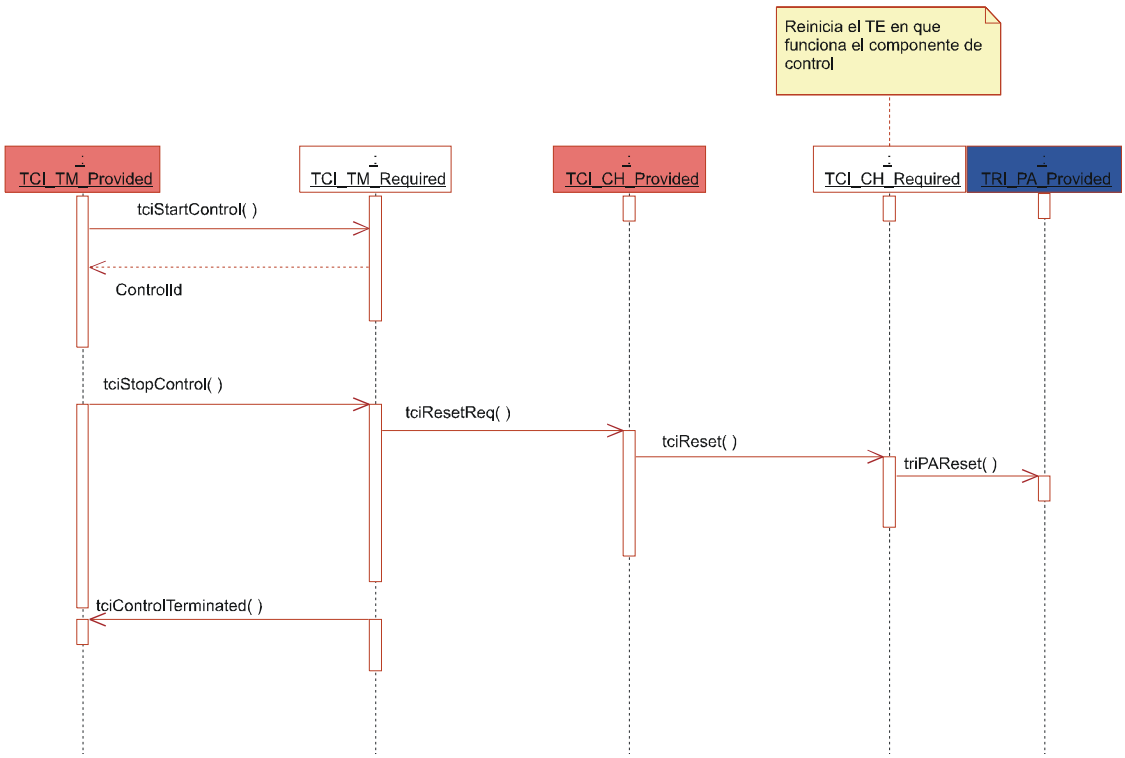

Figura 25/Z.145 – Escenario de utilización – Interrupción de control

11.4.2.2 Fragmento TTCN-3

La interrupción de la parte de control por parte de la gestión de pruebas está fuera del alcance de TTCN-3, por tanto no existe fragmento TTCN-3.

11.4.3 Escenario de utilización: terminación de control después de un error

En la figura 26 se muestra el tratamiento de situaciones de error durante la ejecución de una parte de control, cuando no se está ejecutando ningún caso de prueba. Se informa a la gestión de prueba acerca de la situación de error, teniendo ella entonces que terminar explícitamente la ejecución de la parte de control. Al terminar la parte de control, se reinicializará el sistema de prueba.

11.4.3.1 Diagrama de secuencia

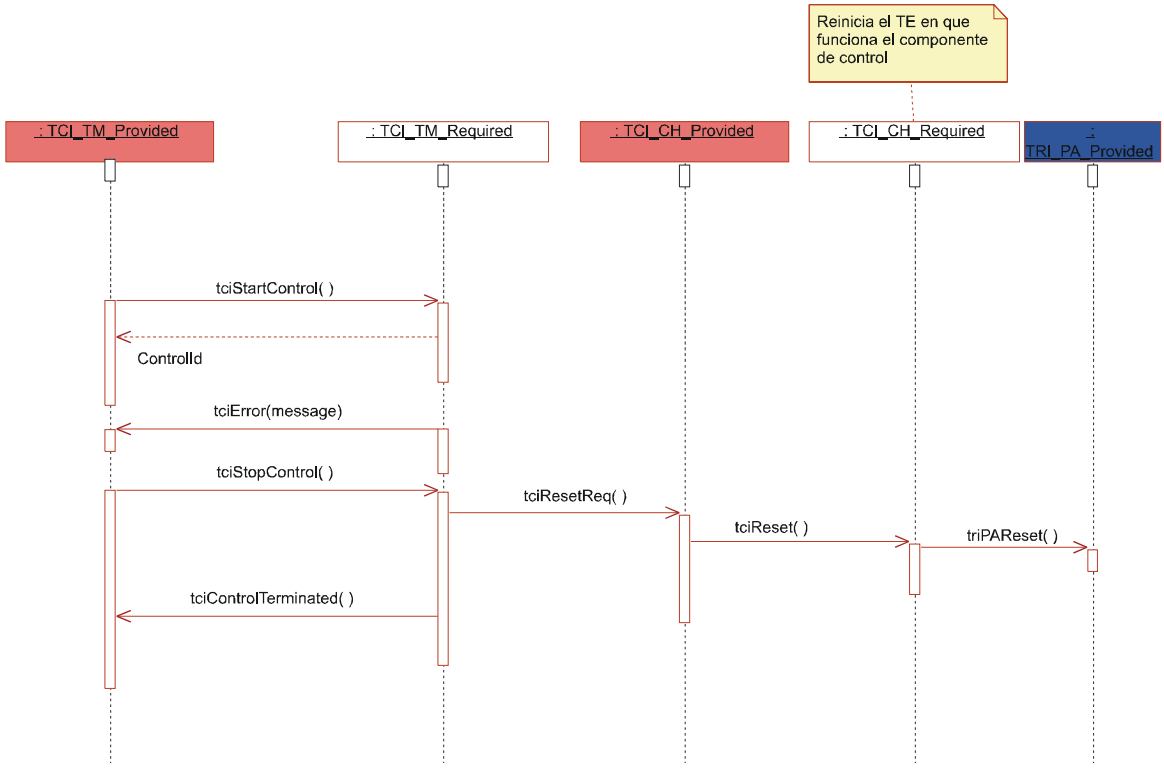

Figura 26/Z.145 – Escenario de utilización – Terminación de control después de un error

11.4.3.2 Fragmento TTCN-3

Las situaciones de error en un sistema de pruebas son excepcionales y no un concepto TTCN-3 en sí mismas, por consiguiente no existe fragmento TTCN-3 para este caso. En su lugar, la semántica TTCN-3 describe diversas situaciones de posibles errores en un sistema de pruebas.

11.4.4 Escenario de utilización: terminación de caso de prueba después de un error

En la figura 27 se describe el tratamiento de situaciones de error durante la ejecución directa de un caso de prueba. Se informa a la gestión de pruebas acerca de la situación de error. La TM debe entonces terminar explícitamente la ejecución del caso de pruebas. Tras la interrupción del caso de pruebas, se interrumpen los componentes paralelos de prueba y se reinicializa el sistema.

11.4.4.1 Diagrama de secuencia

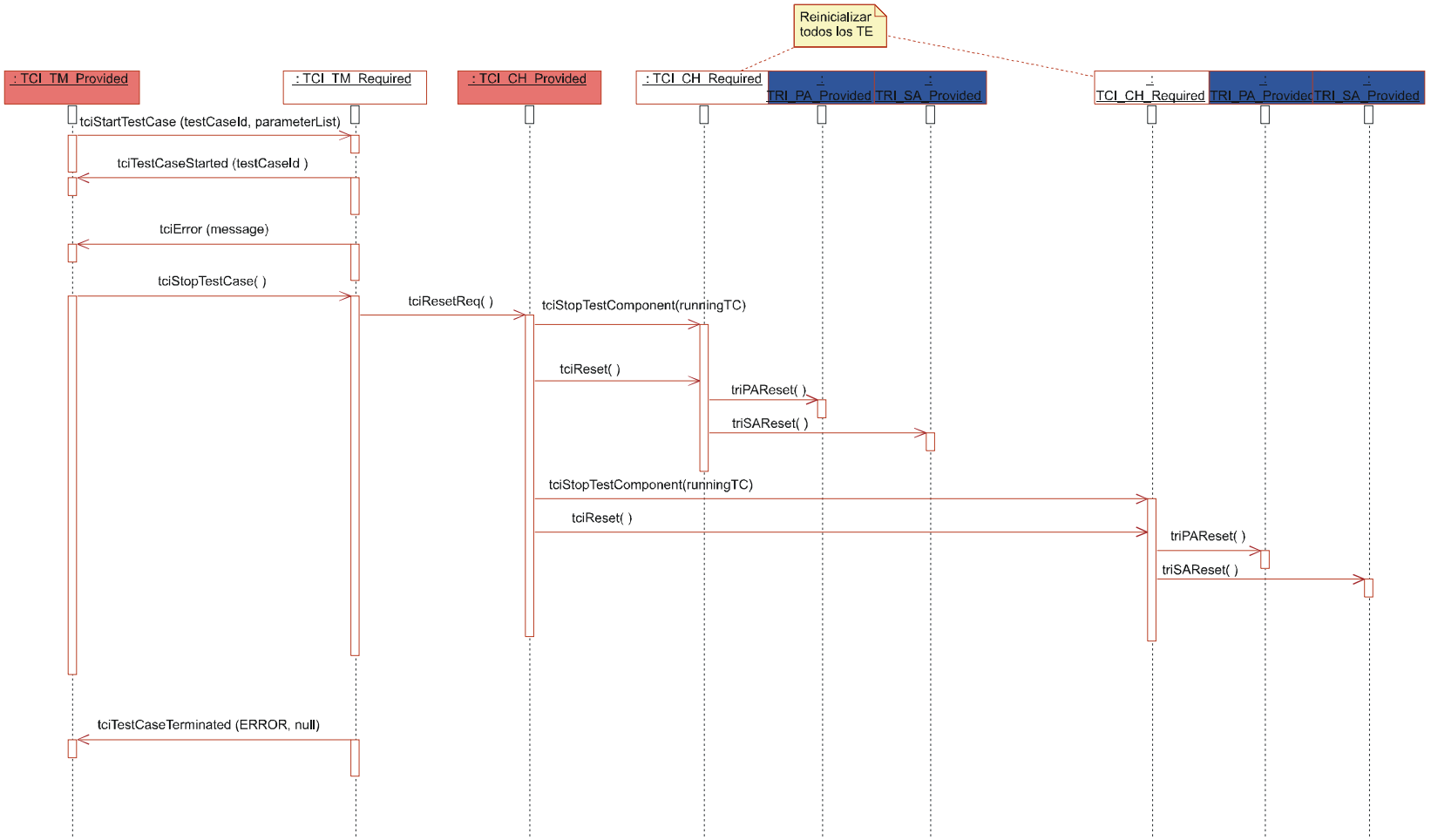

Figura 27/Z.145 – Escenario de utilización – Terminación de caso de prueba después de un error

11.4.4.2 Fragmento TTCN-3

Las situaciones de error en un sistema de pruebas son excepcionales y no un concepto TTCN-3 en sí mismas, por consiguiente no existe fragmento TTCN-3 para este caso. En su lugar, la semántica TTCN-3 describe diversas situaciones de posibles errores en un sistema de pruebas.

11.4.5 Escenario de utilización: reinicialización

En la figura 28 se muestra la reinicialización del sistema de pruebas. En este caso, se reinicializan todos los TE, junto con sus adaptadores de sistema (SA) y adaptadores de plataforma (PA).

11.4.5.1 Diagrama de secuencia

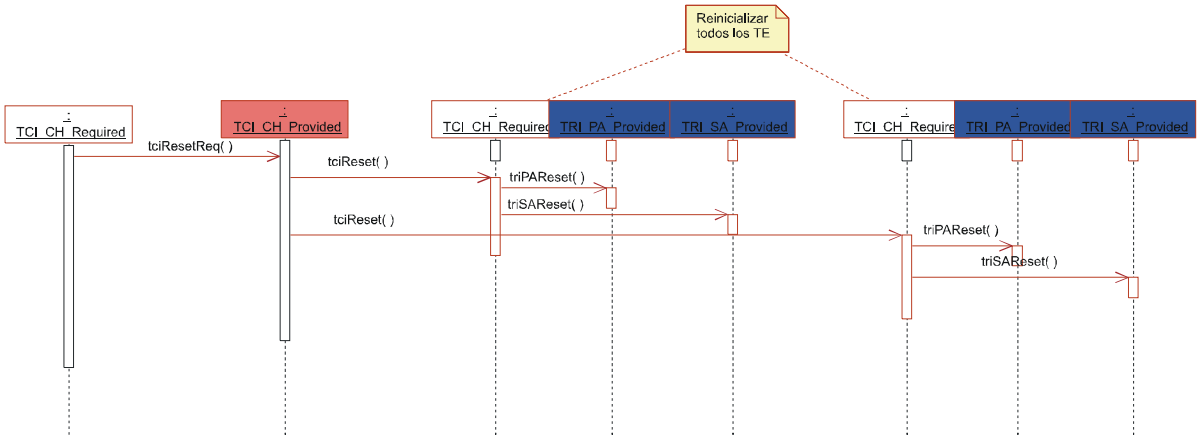

Figura 28/Z.145 – Escenario de utilización – Reinicialización

11.4.5.2 Fragmento TTCN-3

Las reinicializaciones requeridas tras situaciones de error en un sistema de pruebas son excepcionales y no un concepto TTCN-3 en sí mismas, por consiguiente no existe fragmento TTCN-3 para este caso.

11.5 Comunicación

11.5.1 Escenario de utilización: comunicación local entre componentes

En la figura 29 se especifica la comunicación entre componentes de prueba (principales o paralelos) que se encuentran en el mismo nodo. Se pasa una petición de comunicación a la TCI-CH, que decide entonces si pone en cola esta plantilla de comunicación. En este caso, se establece la comunicación localmente a través del TE en el mismo nodo. Se describe aquí una comunicación basada en mensajes que utiliza la operación send (enviar) – el escenario es idéntico para las operaciones call, reply y raise.

11.5.1.1 Diagrama de secuencia

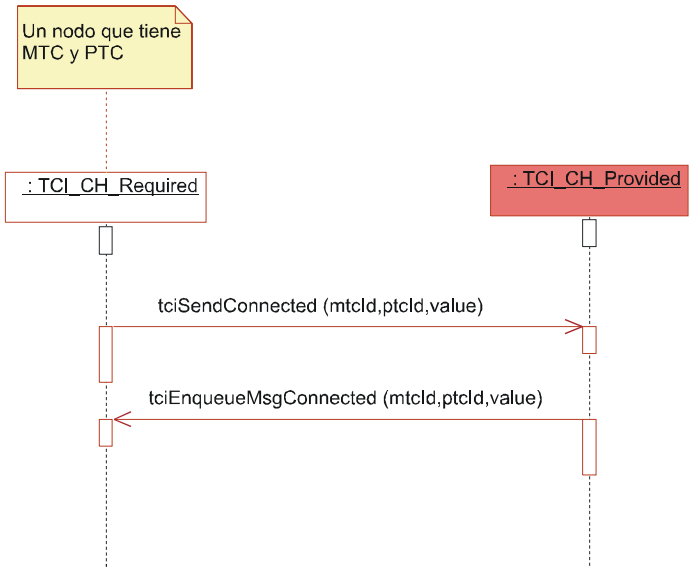

Figura 29/Z.145 – Escenario de utilización – Comunicación local entre componentes

11.5.1.2 Fragmento TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //sending data to a test component
        …
    }
    …
}
```

11.5.2 Escenario de utilización: comunicación internodos entre componentes de prueba

En la figura 30 se especifica la comunicación entre componentes de prueba (principales o paralelos) que se encuentran en diferentes nodos. Se pasa una petición de comunicación a la TCI-CH, que decide entonces si pone en cola esta plantilla de comunicación. En este caso, se establece la comunicación a distancia a través del TE en otro nodo. Se describe aquí una comunicación basada en mensajes que utiliza la operación send (enviar) – el escenario es idéntico para las operaciones call, reply y raise.

11.5.2.1 Diagrama de secuencia

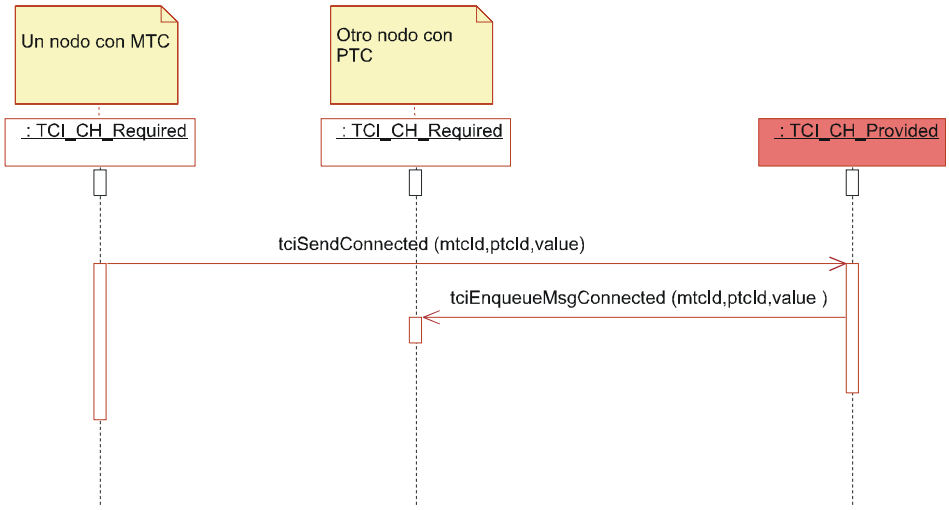

Figura 30/Z.145 – Escenario de utilización – Comunicación internodos entre componentes de prueba

11.5.2.2 Fragmento TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //the PTC behaviour
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //the test case behaviour
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //sending data to a test component
        …
    }
    …
}
```

11.5.3 Escenario de utilización: codificación

En la figura 31 se describe cómo se codifica la información que se envía al SUT. La información codificada se recibe de la entidad de codificación/decodificación a través del TCI-CD. El valor codificado es enviado al SUT a través del TRI-SA. El escenario es idéntico para las operaciones call, reply y raise.

11.5.3.1 Diagrama de secuencia

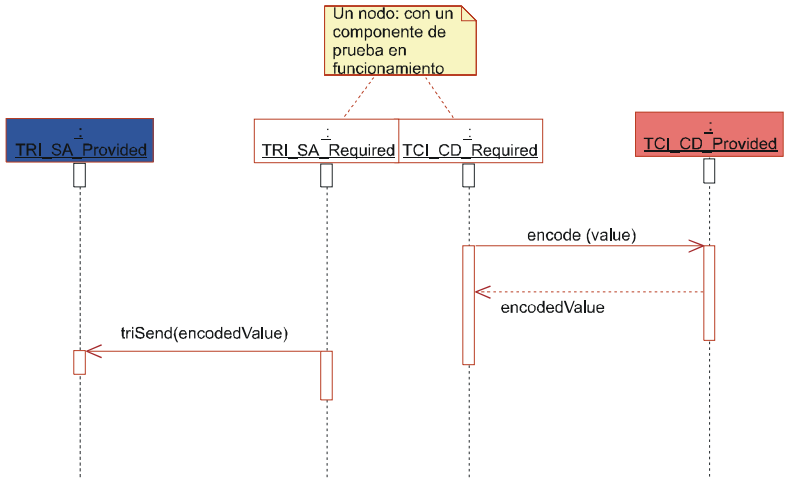

Figura 31/Z.145 – Escenario de utilización – Codificación

11.5.3.2 Fragmento TTCN-3

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

11.5.4 Escenario de utilización: decodificación

En la figura 32 se describe cómo se codifica la información que se recibe del SUT a través del TRI-SA. La información decodificada se recibe de la entidad de codificación/decodificación a través del TCI-CD. El escenario es idéntico para las operaciones receive, getcall, getreply, catch y check.

11.5.4.1 Diagrama de secuencia

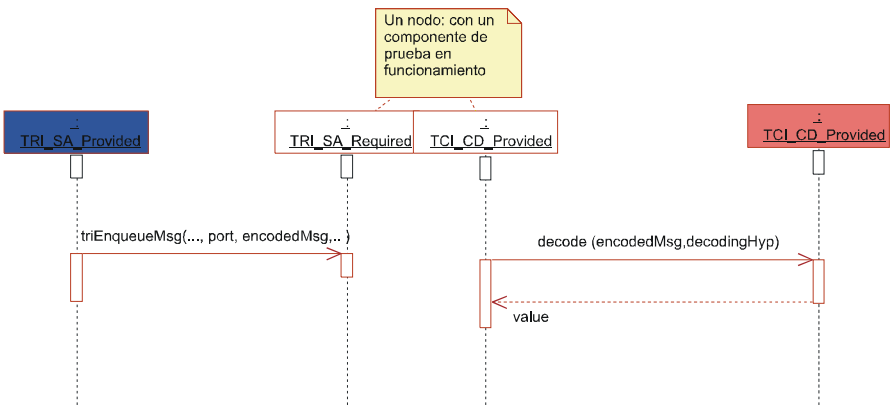

Figura 32/Z.145 – Escenario de utilización – Decodificación

11.5.4.2 Fragmento TTCN-3

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

Anexo A

Especificación IDL de la TCI

En este anexo se definen las interfaces de control TTCN-3 utilizando el lenguaje de definición de interfaz (IDL).

```
// ********************************************************************************
// * Interface definitions for the TTCN-3 Control Interfaces
// ********************************************************************************

module tciInterface {

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

  // ********************************************************************************
  // * Data types taken from the TRI definitions
  // ********************************************************************************

  // Connection
  native TriPortIdType ;
  native TriPortIdListType;
  native TriComponentIdType ;
native TriComponentIdListType;

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

  // Miscellaneous
  native TriStatusType;
  native TriTimerIdType;
native TriTimerDurationType;

  // ********************************************************************************
  // * General Abstract Data Types
  // ********************************************************************************

  // Basic definitions
  native TBoolean;
  native TFloat;
  native TChar;
  native TInteger;
```

```
native TString;
native TUniversalChar;
typedef sequence <TString> TStringSeq;
native TObjid;

struct QualifiedName {
  TString moduleName;
  TString baseName;
};

// General TCI abstract data types
typedef QualifiedName TciBehaviourIdType;
typedef QualifiedName TciModuleIdType;
typedef QualifiedName TciModuleParameterIdType;
typedef QualifiedName TciTestCaseIdType;

enum TciParameterPassingModeType {
  IN_MODE,
  OUT_MODE,
  INOUT_MODE
};

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

typedef sequence <TciParameterType> TciParameterListType;

struct TciParameterTypeType {
  Type parameterType;
  TciParameterPassingModeType mode;
};

typedef sequence <TciParameterTypeType> TciParameterTypeListType;

struct TciModuleParameterType {
  TciModuleParameterIdType parameterName;
  Value defaultValue;
};

typedef sequence <TciModuleIdType> TciModuleIdListType ;

typedef sequence <TciModuleParameterType> TciModuleParameterListType;

typedef sequence <TciTestCaseIdType> TciTestCaseIdListType;


enum TciTestComponentKindType {
  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
};
```

```
// ***************************************************************************
// * Abstract TTCN-3 Data Types And Values
// ***************************************************************************

// Abstract data type "Type"
interface Type {
  TciModuleIdType  getDefiningModule ();
  TString          getName ();
  TciTypeClassType getTypeClass ();
  Value            newInstance ();
  TString          getTypeEncoding ();
  TString          getTypeEncodingVariant ();
  TStringSeq       getTypextension ();
};

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

interface RecordOfValue : Value {
  Value    getField (in TInteger position);
  void     setField (
           in TInteger position,
           in Value value
             );
  void     appendField (in Value value);
  Type     getElementType ();
  TInteger getLength ();
  void     setLength (in TInteger len);
};

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 ();

};

// ***************************************************************************
// * Abstract Logging Types
// ***************************************************************************
```

```
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;


// ********************************************************************************
// Coding Decoding Interface
// -  Required
// ********************************************************************************

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);
};

// ********************************************************************************
// Coding Decoding interface
// -  Provided
// ********************************************************************************

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


// ********************************************************************************
// Test Management Interface
// -  Required
// ********************************************************************************

interface TCI_TM_Required : TCI_CD_Required {
  void tciRootModule (in TciModuleIdType moduleName);
  TciModuleIdListType 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 ();
};

// ********************************************************************************
// Test Management Interface
// -  Provided
// ********************************************************************************

interface TCI_TM_Provided {
  void tciTestCaseStarted (
          in TciTestCaseIdType testCaseId,
          in TciParameterListType parameterList,
          in TFloat timer
          );
  void tciTestCaseTerminated (
          in VerdictValue verdict,
          in TciParameterListType parameterList
          );
  void tciControlTerminated ();
  Value tciGetModulePar (
          in TciModuleParameterIdType parameterId
          );
  void tciLog (
       in TriComponentIdType testComponentId,
       in TString message
       );
  void tciError (in TString message);
};


// ********************************************************************************
// Component Handling Interface
// -  Required
// ********************************************************************************

interface TCI_CH_Required : TCI_CD_Required {
  void tciEnqueueMsgConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in Value receivedMessage
          );
  void tciEnqueueCallConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList
          );
  void tciEnqueueReplyConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in TciParameterListType parameterList,
          in Value returnValue
          );
  void tciEnqueueRaiseConnected (
          in TriPortIdType sender,
          in TriComponentIdType receiver,
          in TriSignatureIdType signature,
          in Value except
          );
  TriComponentIdType tciCreateTestComponent (
          in TciTestComponentKindType kind,
          in Type componentType,
          in 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 ();
};

// *****************************************************************************
// Component Handling Interface
// -  Provided
// *****************************************************************************

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

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

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

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

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

  TriComponentIdType tciCreateTestComponentReq (
          in TciTestComponentKindType kind,
          in Type componentType,
          in 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
          );
};
```

```
// ********************************************************************************
// Test Logging Interface
// -  Provided
// ********************************************************************************

interface TCI_TL_Provided {
  void tliTcExecute(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciTestCaseIdType tcId,
         in 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
           );
    };
};
```

Anexo B

Correspondencia XML para TCI TL proporcionado

En este anexo se define una correspondencia para la interfaz de registro de TCI utilizando definiciones de esquema XML.

B.1 Esquema XML TCI-TL para tipos simples

```
<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">

    <!-- Basic definitions -->
    <xsd:simpleType name="xpath">
        <!-- this string should be XPATH compliant -->
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TBoolean">
        <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>

    <xsd:simpleType name="TString">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TInteger">
        <xsd:restriction base="xsd:integer"/>
    </xsd:simpleType>

    <!-- Miscellaneous -->
    <xsd:simpleType name="TriTimerDurationType">
        <xsd:restriction base="xsd:float"/>
    </xsd:simpleType>

    <xsd:simpleType name="TciParameterPassingModeType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="in"/>
            <xsd:enumeration value="inout"/>
            <xsd:enumeration value="out"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TriStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TRI_Ok"/>
            <xsd:enumeration value="TRI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TciStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TCI_Ok"/>
            <xsd:enumeration value="TCI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

B.2 Esquema XML TCI-TL para tipos

```
<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"/>

<!-- Connection -->
<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>

<!-- Communication -->
<xsd:complexType name="TriMessageType">
    <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>

<xsd:complexType name="TriParameterType">
    <xsd:sequence>
        <xsd:element name="val" type="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>


<!-- Miscellaneous -->
<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>

    <!-- Basic definitions -->
    <xsd:complexType name="QualifiedName">
        <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- general TCI abstract data types -->
    <xsd:complexType name="TciBehaviourIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName" 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>

    <!-- general identifier structure for test components, ports and timer -->
    <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 Esquema XML TCI-TL para valores

```
<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>

<!-- general event elements -->
<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 Esquema XML TCI-TL para plantillas

```
<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 Esquema XML TCI-TL para eventos

```
<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"/>

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

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

    <!-- this event is extended by all port configuration events -->
    <xsd:complexType name="PortConfiguration">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port1" type="Types:TriPortIdType" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="port2" type="Types:TriPortIdType" minOccurs="1"
maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- this event is extended by all port status events -->
    <xsd:complexType name="PortStatus">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- testcases -->
    <xsd:complexType name="tliTcExecute">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="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>

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

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

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

    <!-- asynchronous communication -->
    <xsd:complexType name="tliMSend_m">
         <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="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>

     <!-- synchronous communication -->
    <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>

<!-- components -->
<xsd:complexType name="tliCCreate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="name" type="SimpleTypes:TString"/>
            </xsd: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>

<!-- ports -->
<xsd:complexType name="tliPConnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPDisconnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPMap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPUnmap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPClear">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events: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>

<!-- 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>

    <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>

    <!-- timers -->
    <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>

    <!-- scope -->
    <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>

    <!-- variables and module parameter -->
    <xsd:complexType name="tliVar">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" 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>
```

```
    <!-- verdicts -->
    <xsd:complexType name="tliGetVerdict">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="verdict" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliSetVerdict">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="verdict" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- log -->
    <xsd:complexType name="tliLog">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="log" type="SimpleTypes:TString"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

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

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

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

    <xsd:complexType name="tliAActivate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" 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 Esquema XML TCI-TL para un registro

```
<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">

            <!-- test cases operations -->
            <xsd:element name="tliTcExecute" type="Events:tliTcExecute"/>
            <xsd:element name="tliTcStart" type="Events:tliTcStart"/>
            <xsd:element name="tliTcStop" type="Events:tliTcStop"/>
            <xsd:element name="tliTcStarted" type="Events:tliTcStarted"/>
            <xsd:element name="tliTcTerminated" type="Events:tliTcTerminated"/>

            <!-- control operations -->
            <xsd:element name="tliCtrlStart" type="Events:tliCtrlStart"/>
            <xsd:element name="tliCtrlStop" type="Events:tliCtrlStop"/>
            <xsd:element name="tliCtrlTerminated" type="Events:tliCtrlTerminated"/>

            <!-- asynchronous communication -->
            <xsd:element name="tliMSend_m" type="Events:tliMSend_m"/>
            <xsd:element name="tliMSend_c" type="Events:tliMSend_c"/>
            <xsd:element name="tliMDetected_m" type="Events:tliMDetected_m"/>
            <xsd:element name="tliMDetected_c" type="Events:tliMDetected_c"/>
            <xsd:element name="tliMMismatch_m" type="Events:tliMMismatch_m"/>
            <xsd:element name="tliMMismatch_c" type="Events:tliMMismatch_c"/>
            <xsd:element name="tliMReceive_m" type="Events:tliMReceive_m"/>
            <xsd:element name="tliMReceive_c" type="Events:tliMReceive_c"/>

            <!-- synchronous communication -->
            <xsd:element name="tliPrCall_m" type="Events:tliPrCall_m"/>
            <xsd:element name="tliPrCall_c" type="Events:tliPrCall_c"/>
```

```
<xsd:element name="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"/>

<!-- components -->
<xsd:element name="tliCCreate" type="Events:tliCCreate"/>
<xsd:element name="tliCStart" type="Events:tliCStart"/>
<xsd:element name="tliCRunning" type="Events:tliCRunning"/>
<xsd:element name="tliCAlive" type="Events: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"/>

<!-- ports -->
<xsd:element name="tliPConnect" type="Events:tliPConnect"/>
<xsd:element name="tliPDisconnect" type="Events:tliPDisconnect"/>
<xsd:element name="tliPMap" type="Events:tliPMap"/>
<xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>
<xsd:element name="tliPClear" type="Events:tliPClear"/>
<xsd:element name="tliPStart" type="Events:tliPStart"/>
<xsd:element name="tliPStop" type="Events:tliPStop"/>
<xsd:element name="tliPHalt" type="Events:tliPStop"/>

<!-- codec -->
<xsd:element name="tliDecode" type="Events:tliDecode"/>
<xsd:element name="tliEncode" type="Events:tliEncode"/>

<!-- timers -->
<xsd:element name="tliTTimeoutDetected" type="Events:tliTTimeoutDetected"/>
<xsd:element name="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"/>

<!-- scopes -->
<xsd:element name="tliSEnter" type="Events:tliSEnter"/>
<xsd:element name="tliSLeave" type="Events:tliSLeave"/>

<!-- statements -->
<xsd:element name="tliVar" type="Events:tliVar"/>
<xsd:element name="tliGetVerdict" type="Events:tliGetVerdict"/>
<xsd:element name="tliSetVerdict" type="Events:tliSetVerdict"/>
<xsd:element name="tliLog" type="Events:tliLog"/>

<!-- alt  -->
<xsd:element name="tliAEnter" type="Events:tliAEnter"/>
<xsd:element name="tliALeave" type="Events:tliALeave"/>
<xsd:element name="tliADefaults" type="Events:tliADefaults"/>
<xsd:element name="tliAActivate" type="Events:tliAActivate"/>
```

```
            <xsd:element name="tliADeactivate" type="Events:tliADeactivate"/>
            <xsd:element name="tliANomatch" type="Events:tliANomatch"/>
            <xsd:element name="tliARepeat" type="Events:tliARepeat"/>
            <xsd:element name="tliAWait" type="Events:tliAWait"/>

        </xsd:choice>
    </xsd:complexType>
</xsd:schema>
```

BIBLIOGRAFÍA

- INTOOL CGI/NPL038 (V2.2): *Generic Compiler/Interpreter interface; GCI Interface Specification Infrastructural Tools for Informational Technology and Telcomunications Conference Testing*, diciembre de 1996.
- Recomendación UIT-T X.292 (2002), *Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos de las Recomendaciones sobre los protocolos para aplicaciones del UIT-T – Notación combinada arborescente y tabular*.

  ISO/CEI 9646-3:1998, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 3*: *The Tree and Tabular combined Notation* (*TTCN*).
- ISO/CEI 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, febrero de 1998.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedia
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedia
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	Gestión de las telecomunicaciones, incluida la RGT y el mantenimiento de redes
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	Terminales para servicios de telemática
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos, comunicaciones de sistemas abiertos y seguridad
Serie Y	Infraestructura mundial de la información, aspectos del protocolo Internet y Redes de la próxima generación
Serie Z	**Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación**

Impreso en Suiza
Ginebra, 2007