Unión Internacional de Telecomunicaciones

# UIT-T Z.144

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

(03/2006)

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 ejecución

Recomendación UIT-T Z.144

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.144**

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


## Resumen

Esta Recomendación contiene la especificación de la interfaz de ejecución para las implementaciones del sistema de prueba TTCN-3. La interfaz de ejecución TTCN-3 proporciona la adaptación recomendada para la temporización y la comunicación de un sistema de prueba con una determinada plataforma de procesamiento y con el sistema que es objeto de prueba, respectivamente. En esta Recomendación se define la interfaz como un conjunto de operaciones independientes del lenguaje elegido.

La definición de esta interfaz es compatible con la Rec. UIT-T Z.140. En esta Recomendación se utiliza el lenguaje de definición de interfaz (IDL) CORBA para especificar la TRI completamente. En las cláusulas 6 y 7 se especifican correspondencias lingüísticas de la especificación abstracta con los lenguajes Java y ANSI-C. El anexo A contiene un resumen de la especificación de interfaz basada en el IDL.



## Orígenes

La Recomendación UIT-T Z.144 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/dbase/patent/index.html.

**ÍNDICE**

*Página*


1 Alcance ..... 1
1.1 Conformidad ..... 1
2 Referencias ..... 1
3 Definiciones y abreviaturas ..... 2
3.1 Definiciones ..... 2
3.2 Abreviaturas, siglas o acrónimos ..... 3
4 Estructura general de un sistema de prueba TTCN-3 ..... 3
4.1 Entidades de un sistema de prueba TTCN-3 ..... 4
4.2 Interfaces en un sistema de prueba TTCN-3 ..... 6
4.3 Requisitos de ejecución de un sistema de prueba TTCN-3 ..... 7
5 Operaciones e interfaz de ejecución TTCN-3 ..... 7
5.1 Panorama general de la TRI ..... 7
5.2 Gestión de errores ..... 9
5.3 Interfaz de datos ..... 9
5.4 Descripción de operaciones ..... 10
5.5 Operaciones de interfaz de comunicaciones ..... 11
5.6 Operaciones de interfaz de plataformas ..... 23
6 Correspondencia con el lenguaje Java ..... 25
6.1 Introducción ..... 25
6.2 Nombres y alcances ..... 25
6.3 Correspondencia de tipos ..... 26
6.4 Constantes ..... 33
6.5 Correspondencia de interfaces ..... 33
6.6 Parámetros opcionales ..... 36
6.7 Inicialización de la TRI ..... 36
6.8 Gestión de errores ..... 36
7 Correspondencia con el lenguaje ANSI-C ..... 37
7.1 Introducción ..... 37
7.2 Nombres y alcances ..... 37
7.3 Gestión de memoria ..... 41
7.4 Gestión de errores ..... 41
8 Escenarios de utilización ..... 41
8.1 Primer escenario ..... 42
8.2 Segundo escenario ..... 44
8.3 Tercer escenario ..... 46
Anexo A (normativo) – Resumen del IDL ..... 48
BIBLIOGRAFÍA ..... 51

## Introducción

Esta Recomendación consta de dos partes distintas; en la primera parte se describe la estructura de la implementación de un sistema de prueba TTCN-3 y en la segunda parte se presenta la especificación de la interfaz en tiempo de ejecución TTCN-3.

En la primera parte se describe la descomposición de un sistema de prueba TTCN-3 en cuatro entidades principales:

- gestión de prueba (TM);
- TTCN-3 ejecutable (TE);
- adaptador SUT (SA); y
- adaptador de plataforma (PA).

Se define asimismo la interacción entre esas entidades, es decir, las correspondientes interfaces.

En la segunda parte de esta Recomendación se especifica la interfaz de ejecución TTCN-3 (TRI). Esa interfaz se define sobre la base de las operaciones, las cuales se realizan como parte de una entidad y son llamadas por otras entidades del sistema de prueba. La especificación de la interfaz define para cada operación las estructuras de datos conexas, el efecto previsto del sistema de prueba y cualesquiera limitaciones en cuanto al uso de la operación. Cabe señalar que esta especificación de interfaz sólo define interacciones entre la TSI y el SUT, así como las operaciones del temporizador.

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

## 1 Alcance

Esta Recomendación especifica la interfaz de ejecución para las implementaciones del sistema de prueba TTCN-3. La interfaz de ejecución TTCN-3 es una adaptación normalizada para la temporización y la comunicación de un sistema de prueba con una plataforma de procesamiento determinada y con el sistema que es objeto de prueba, respectivamente. En esta Recomendación se define la interfaz como una serie de operaciones independientes del lenguaje elegido.

La definición de esta interfaz es compatible con la norma TTCN-3 (véase la referencia *infra*). En esta Recomendación se utiliza el lenguaje de definición de interfaz (IDL, *interface definition language*) CORBA para especificar la TRI completamente. Las cláusulas 6 y 7 contienen correspondencias lingüísticas de esta especificación abstracta con los lenguajes Java y ANSI-C. El anexo A contiene un resumen de la especificación de interfaz basada en el IDL.

### 1.1 Conformidad

El requisito de que un sistema de prueba TTCN-3 sea conforme con la TRI significa que éste debe cumplir con la especificación de interfaz definida en esta Recomendación, así como con una de las correspondencias de lenguaje previstas.

EJEMPLO: Si un vendedor soporta el lenguaje Java, las implementaciones y llamadas a operaciones TRI, que forman parte de la TTCN-3 ejecutable, deben estar en conformidad con el IDL para la correspondencia Java especificada en esta Recomendación.

## 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 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.*

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

[3] Recomendación UIT-T X.292 (2002), *Metodología y marco de las pruebas de conformidad para interconexión de sistemas abiertos en 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).*

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

# 3 Definiciones y abreviaturas

## 3.1 Definiciones

A los efectos de la presente Recomendación, se aplican los términos y definiciones consignados en la Rec. UIT-T Z.140 [2], además de los siguientes:

**3.1.1 sucesión de pruebas abstractas (ATS, *abstract test suite*)**: Véase Rec. UIT-T X.290 [1].

**3.1.2 puerto de comunicación**: Mecanismo abstracto que facilita la comunicación entre los componentes de prueba.

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

**3.1.3 sucesión de pruebas ejecutables (ETS, *executable test suite*)**: Véase Rec. UIT-T X.290 [1].

**3.1.4 temporizador explícito**: Temporizador declarado en una TTCN-3 ATS y al cual se puede acceder a través de las operaciones del temporizador TTCN-3.

**3.1.5 información suplementaria de implementación para pruebas (IXIT, *implementation extra information for testing*)**: Véase Rec. UIT-T X.290 [1].

**3.1.6 temporizador implícito**: Temporizador del sistema que es creado por la TTCN-3 ejecutable para guardar una llamada TTCN-3 o ejecutar operaciones.

NOTA – El usuario de la TTCN-3 no puede acceder a los temporizadores implícitos.

**3.1.7 adaptador de plataforma (PA, *platform adapter*)**: Entidad que adapta la TTCN-3 ejecutable a una plataforma de ejecución determinada.

NOTA – El adaptador de plataforma crea una noción única de tiempo para un sistema de prueba TTCN-3 e implementa las funciones externas, así como los temporizadores explícito e implícito.

**3.1.8 adaptador SUT (SA, *SUT adapter*)**: Entidad que adapta las operaciones de comunicación TTCN-3 con el SUT sobre la base de una interfaz de sistema de pruebas abstractas e implementa la interfaz del sistema de prueba real.

**3.1.9 sistema sometido a prueba (SUT, *system under test*)**: Véase Rec. UIT-T X.290 [1].

NOTA – Todos los tipos son conocidos en el momento de la compilación, es decir, son estáticos.

**3.1.10 caso de prueba**: Véase Rec. UIT-T X.290 [1].

**3.1.11 evento de prueba**: Datos de prueba enviados o recibidos (mensajes o llamadas a procedimientos) en un puerto de comunicación que forma parte de la interfaz del sistema de prueba.

**3.1.12 gestión de prueba (TM, *test management*)**: Entidad que proporciona una interfaz de usuario y administra el sistema de prueba TTCN-3.

**3.1.13 sistema de prueba**: Véase Rec. UIT-T X.290 [1].

**3.1.14 interfaz de sistema de prueba**: Componente de prueba que proporciona una correspondencia de los puertos disponibles en el sistema de prueba TTCN-3 (abstracto) con los ofrecidos por un sistema de prueba real.

**3.1.15 identificación de temporizador (TID, *timer identification*)**: Identificación única para ejemplares de temporizador explícito o implícito que genera la TTCN-3 ejecutable.

**3.1.16 interfaz de control TTCN-3 (TCI, *TTCN-3 control interface*)**: Actualmente una interfaz patentada que especifica la interacción entre la gestión de prueba y la TTCN-3 ejecutable en un sistema de prueba.

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

**3.1.18 interfaz de ejecución TTCN-3 (TRI, *TTCN-3 runtime interface*)**: Interfaz que define la interacción de la TTCN-3 ejecutable con el SUT y el adaptador de plataforma en un sistema de prueba.

### 3.2 Abreviaturas, siglas o acrónimos

En esta Recomendación se utilizan las siguientes abreviaturas, siglas o acrónimos.

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

CH Manipulador de componentes (*component handler*)

ECD CóDecs externos (*external CoDecs*)

EDS Sistema de codificación/decodificación (*encoding/decoding system*)

ETS Sucesión de prueba ejecutable (*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 secuencia de mensajes (*message sequence chart*)

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

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

PA Adaptador de plataforma (*platform adapter*)

SA Adaptador SUT (*SUT adapter*)

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

T3RTS Sistema de ejecución TTCN-3 (*TTCN-3 runtime system*)

TC Control de prueba (*test control*)

TCI Interfaz de control TTCN-3 (*TTCN-3 control interface*)

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

TID Identificación de temporizador (*timer identification*)

TL Entrada al sistema de prueba (*test logging*)

TM Gestión de la prueba (*test management*)

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

TSI Interfaz del sistema de prueba (*test system interface*)

TTCN Notación de pruebas y de control de pruebas (*testing and test control notation*)

TTCN-3 Notación combinada arborescente y tabular versión 3 (*tree and tabular combined notation version 3*)

## 4 Estructura general de un sistema de prueba TTCN-3

Conceptualmente un sistema de prueba TTCN-3 puede considerarse como un conjunto de entidades que interactúan, en el cual cada entidad corresponde a un aspecto particular de la funcionalidad en la implementación de un sistema de prueba. Esas entidades gestionan la ejecución de la prueba, interpretan o ejecutan el código TTCN-3 compilado, realizan comunicaciones adecuadas con el SUT, efectúan funciones externas y manipulan las operaciones del temporizador. (Véase la figura 1.)

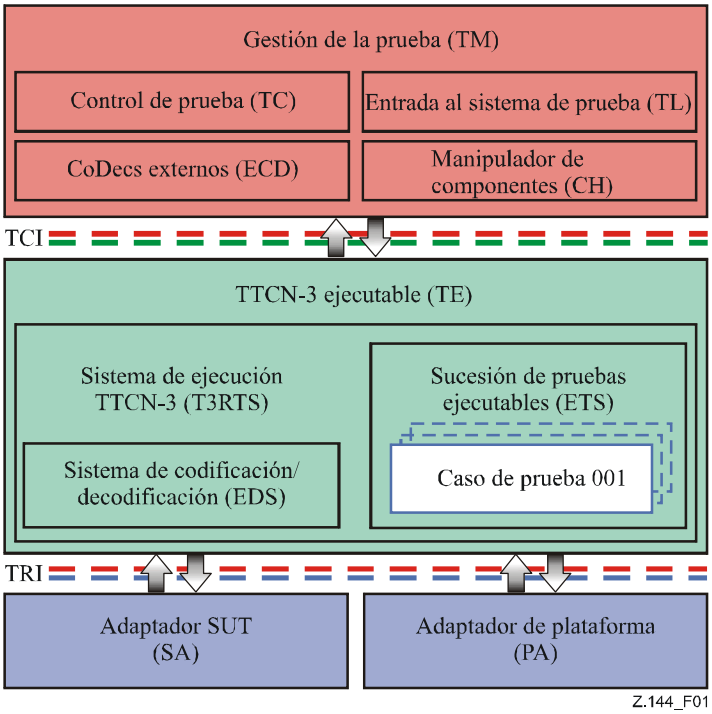


**Figura 1/Z.144 – Estructura general de un sistema de prueba TTCN-3**

## 4.1 Entidades de un sistema de prueba TTCN-3

En la figura 1 se ilustra la estructura de la implementación de un sistema de prueba TTCN-3. Cabe señalar que el ulterior perfeccionamiento de TM en entidades más pequeñas, según se muestra en la figura 1 y se utiliza en las siguientes cláusulas de esta Recomendación, no es nada más que una ayuda para definir las interfaces del sistema de prueba TTCN-3.

La parte del sistema de prueba que trata de la interpretación y ejecución de los módulos TTCN-3, es decir, la sucesión de pruebas ejecutables (ETS), forma parte de la TTCN-3 ejecutable (TE). Esto corresponde ya sea al código ejecutable producido por un compilador TTCN-3 o a un interpretador TTCN-3 en una implementación del sistema de prueba. Se supone que la implementación de un sistema de prueba incluye el ETS, obtenido a partir de la TTCN-3 ATS.

La parte restante del sistema de prueba TTCN-3, que trata de cualquier aspecto que no pueda determinarse a partir de la información contenida en el ATS original únicamente, puede descomponerse en las entidades gestión de prueba (TM), adaptador SUT (SA) y adaptador de plataforma (PA). En general estas entidades abarcan una interfaz de usuario del sistema de prueba, el control de la ejecución de prueba, la entrada al sistema del evento de prueba, así como la comunicación con el SUT y la implementación del temporizador.

### 4.1.1 Gestión de prueba (TM)

En la entidad TM podemos hacer una distinción entre la funcionalidad relacionada con el control de ejecución de la prueba y la entrada al sistema del evento de prueba.

#### 4.1.1.1 Control de prueba (TC)

La entidad TC es responsable de la gestión general del sistema de prueba. Después de que el sistema de prueba ha sido inicializado, la ejecución de la prueba comienza dentro de la entidad TC. Esta entidad es responsable de la invocación adecuada de los módulos TTCN-3, es decir, de la propagación de los parámetros del módulo y/o de la información IXIT a la TE, en caso necesario. Por lo general esta entidad también implementará una interfaz de usuario del sistema de prueba.

#### 4.1.1.2 Entrada al sistema de prueba (TL)

La entidad TL es responsable de mantener el registro de la prueba. La TE notifica explícitamente que se registren los eventos de prueba. La entidad TL tiene una interfaz unidireccional en la cual cualquier entidad que forme parte de la TE puede colocar una solicitud de entrada al sistema TL. También podría utilizarse una interfaz interna TM para registrar la información de gestión de prueba generada por la entidad TC.

#### 4.1.1.3 Códecs externos (ECD)

Las entidades Códecs externos son facultativamente responsables de codificar y decodificar los datos relacionados con las comunicaciones basadas en mensajes o basadas en procedimientos dentro del TE. Los códecs externos pueden utilizarse junto con los códecs incorporados en la entidad TE, o en lugar de los mismos. A diferencia de los códecs incorporados, los códecs externos tienen una interfaz normalizada, gracias a lo cual pueden ser portátiles entre diferentes instrumentos y sistemas TTCN-3.

#### 4.1.1.4 Manipulador de componentes (CH)

La entidad CH es responsable de distribuir los componentes de prueba en paralelo. Esa distribución podría efectuarse a través de uno o numerosos sistemas físicos. La entidad CH permite a la gestión de prueba crear y controlar sistemas de prueba especializados de una manera transparente e independiente de la entidad TE.

### 4.1.2 TTCN-3 ejecutable (TE)

La entidad TE es responsable de la interpretación o ejecución de la TTCN-3 ATS. En términos conceptuales, la entidad TE puede descomponerse en tres entidades interactuantes: la ETS, el sistema de ejecución TTCN-3 (T3RTS) y el sistema de codificación/decodificación (EDS, *encoding/decoding system*). Cabe señalar que este perfeccionamiento de la entidad TE en entidades más pequeñas es estrictamente una ayuda conceptual para definir las interfaces del sistema de prueba TTCN-3; no es necesario que esta distinción se refleje en las implementaciones de la TRI.

En las cláusulas siguientes se definen las responsabilidades de cada entidad y se examina asimismo la manipulación de los temporizadores en la TRI.

#### 4.1.2.1 Sucesión de pruebas ejecutables (ETS)

La entidad ETS gestiona la ejecución o interpretación de casos de prueba, la disposición en secuencias y la correspondencia de los eventos de prueba, según se definen en los correspondientes módulos TTCN-3 de la Rec. UIT-T Z.140 [2]. Esta entidad interactúa con la entidad T3RTS para enviar, tratar de recibir (o hacer corresponder) y registrar eventos de prueba durante la ejecución de un caso de prueba, para crear o suprimir componentes de prueba TTCN-3, así como para manipular llamadas a funciones externas, operaciones de acción y temporizadores. Cabe señalar que la entidad ETS no interactúa directamente con el SA a través de la TRI.

#### 4.1.2.2 Sistema de ejecución TTCN-3 (T3RTS)

La entidad T3RTS interactúa con las entidades TM, SA y PA por conducto de las entidades TCI y TRI, y gestiona a las entidades ETS y EDS. El sistema T3RTS inicializa los adaptadores, así como las entidades ETS y EDS. Esta entidad realiza todas las acciones necesarias para iniciar adecuadamente la ejecución de una función o caso de prueba con parámetros en la entidad ETS. Ésta consulta a la entidad TM acerca de los valores de parámetros de módulo requeridos por la ETS y le envía a ésta información de registro; asimismo, compila y resuelve los veredictos conexos devueltos por la entidad ETS, según se define en la Rec. UIT-T Z.140 [2].

La entidad T3RTS implementa la creación y supresión de componentes de prueba TTCN-3, así como la semántica TTCN-3 de comunicaciones basadas en mensajes y procedimientos, llamadas a funciones externas, operaciones de acción y temporizadores. Esto incluye notificar al adaptador SUT (SA) cuál es el mensaje o la llamada a un procedimiento que se debe enviar al SUT, o al adaptador de plataforma (PA), cuál es la función que se debe ejecutar o qué temporizadores se deben inicializar, parar, consultar o leer. Análogamente, T3RTS notifica a la entidad ETS los mensajes o llamadas a procedimientos entrantes procedentes del SUT, así como eventos de expiración.

Antes de enviar o recibir mensajes y llamadas a procedimientos hacia o desde el SA, o de manipular las llamadas a funciones y las operaciones de acción en el PA para la entidad ETS, T3RTS invoca a la entidad EDS para su codificación o decodificación. La entidad T3RTS debería implementar todas las operaciones de comunicaciones basadas en mensajes y procedimientos entre los componentes de prueba, pero sólo la semántica TTCN-3 de comunicaciones basadas en procedimientos con el SUT, es decir, el posible bloqueo y desbloqueo de la ejecución de componentes de prueba, la supervisión de los temporizadores implícitos y la manipulación de las excepciones de expiración como resultado de esas operaciones de comunicación. Todas las operaciones de comunicación con el SUT basadas en procedimientos deben realizarse e identificarse (en el caso de una operación receptora) en el SA, ya que éstas se implementan de manera más eficaz en una plataforma específica. Cabe señalar que la temporización de cualquier operación de llamada a un procedimiento, por ejemplo los temporizadores implícitos, se implementa en el adaptador de plataforma (PA).

Es necesario que la TTCN-3 ejecutable mantenga sus propias colas de puerto (distintas de las que podrían estar disponibles en el SA o el PA) con el fin de que los eventos de prueba de entrada efectúen instantáneas para recibir las operaciones definidas en la Rec. UIT-T Z.140 [2]. Los eventos de expiración, que son generados por las implementaciones del temporizador TTCN-3, el temporizador de llamadas o el temporizador de casos de prueba, deben mantenerse en una lista de expiración, según se especifica en la Rec. UIT-T X.292 [3]. En la figura 2 todas estas

funcionalidades se han asignado a la entidad T3RTS. Ésta es responsable de almacenar los eventos que el SA o el PA hayan notificado a la entidad TE, pero que aún deben ser procesados.

#### 4.1.2.3 Sistema de codificación/decodificación (EDS)

La entidad EDS es responsable de la codificación y decodificación de los datos de prueba, que incluyen los datos utilizados en operaciones de comunicación con el SUT, según se especifica en el módulo TTCN-3 ejecutante. Si no se ha especificado ninguna codificación para un módulo TTCN-3, la codificación de los valores de datos depende del instrumento. Esta entidad es invocada por la entidad T3RTS y devuelta a la misma. Cabe señalar que la entidad EDS no interactúa directamente con el SA a través de la TRI.

#### 4.1.2.4 Temporizadores en la TTCN-3 ejecutable

Los temporizadores que han sido declarados y nombrados en la TTCN-3 ATS pueden clasificarse conceptualmente como explícitos en la entidad TE. Los temporizadores creados por TE para supervisar las llamadas a procedimientos TTCN-3 o ejecutar operaciones se conocen en la TE como temporizadores implícitos. Tanto los temporizadores explícitos como los implícitos son creados dentro de la TE pero implementados por el adaptador de plataforma (PA). Esto se logra generando una identificación de temporizador (TID, *timer IDentification*) única para cada temporizador creado en la TE. Esta TID única debe permitir a la TE hacer una distinción entre diferentes temporizadores. La TE utiliza el TID para interactuar con la correspondiente implementación de temporizador en PA.

Cabe señalar que la TE tiene la responsabilidad de implementar las diferentes semánticas TTCN-3 para temporizadores explícitos e implícitos correctamente, según se define en la Rec. UIT-T Z.140 [2]; por ejemplo, el uso de las palabras clave `any` y `all` con temporizadores sólo se aplica a los temporizadores explícitos. En PA todos los temporizadores, ya sean implícitos o explícitos, se tratan de la misma manera.

### 4.1.3 Adaptador SUT (SA)

El SA adapta la comunicación basada en mensajes y procedimientos del sistema de prueba TTCN-3 con el SUT a una plataforma de ejecución determinada del sistema de prueba. Éste es consciente de la correspondencia de los puertos de comunicación del componente de prueba TTCN-3 con los puertos de la interfaz del sistema de prueba, e implementa la interfaz del sistema de prueba real, como se define en la Rec. UIT-T Z.140 [2]. El SA es responsable de propagar solicitudes de envío y operaciones de acción desde la TTCN-3 ejecutable (TE) hasta el SUT, y de notificar a la TE acerca de cualquier evento de prueba recibido, anexando dichos eventos a las colas del puerto de la TE.

Las operaciones de comunicación con el SUT basadas en procedimientos se implementan en el SA. Este último es responsable de hacer una distinción entre los diferentes mensajes dentro de las comunicaciones basadas en procedimientos (por ejemplo, llamada, respuesta y excepción) y propagar dichos mensajes de la manera adecuada, ya sea a la SUT o a la TE. La semántica de la comunicación basada en procedimientos TTCN-3, es decir, el efecto de dicha operación en la ejecución del componente de prueba TTCN-3, se ha de manipular en el TE.

El SA tiene una interfaz con la TE, que se utiliza para enviar mensajes SUT (emitidos en operaciones de acción TTCN-3 SUT) al SA e intercambiar datos de prueba codificados entre las dos entidades en las operaciones de comunicación con el SUT.

### 4.1.4 Adaptador de plataforma (PA)

El PA implementa las funciones externas TTCN-3 y proporciona un sistema de prueba TTCN-3 con una noción única de tiempo. En esa entidad se han de realizar las funciones externas, además de implementar todos los temporizadores. Cabe señalar que en el TE se crean ejemplares de temporizador. En el PA, un temporizador sólo puede distinguirse por su identificación de temporizador (TID). Por lo tanto, el PA trata de igual modo a los temporizadores explícito e implícito.

La interfaz con la TE permite la invocación de funciones externas y la iniciación, lectura y parada de los temporizadores, así como la consulta sobre el estado de los temporizadores utilizando su ID de temporizador. El PA notifica a la TE acerca de los temporizadores expirados.

## 4.2 Interfaces en un sistema de prueba TTCN-3

Según se indicó anteriormente en la figura 1, un sistema de prueba TTCN-3 tiene dos interfaces, a saber, la interfaz de control TTCN-3 (TCI) y la interfaz de ejecución TTCN-3 (TRI), las cuales especifican la interfaz entre las entidades gestión de prueba (TM) y TTCN-3 ejecutable (TE) y las entidades TE, adaptador SUT (SA) y adaptador de plataforma (PA), respectivamente.

En esta Recomendación se define la TRI. La interacción de la TE con el SA y el PA se definirán en función de las operaciones TRI. Aunque ambas interfaces, es decir, TRI y TCI, deben definirse para una implementación completa de

un sistema de prueba TTCN-3, actualmente se considera que la especificación y la implementación de la TCI está patentada.

### 4.3 Requisitos de ejecución de un sistema de prueba TTCN-3

En la entidad llamante, cada llamada a una operación TRI se considerará como una operación atómica. La entidad llamada, que implementa la operación TRI, devolverá el control a la entidad llamante tan pronto como haya logrado su efecto previsto o en caso de que la operación no pueda terminarse con éxito. La entidad llamada no se bloqueará al realizar la comunicación basada en procedimientos. No obstante, dicha entidad se bloqueará después de invocar una implementación de función externa y esperar su valor de retorno. Cabe señalar que si durante la implementación del sistema de prueba no se logra volver de una implementación de función externa, se podría provocar el bloqueo infinito de la ejecución del componente de prueba, la TTCN-3 ejecutable, el adaptador de plataforma o, incluso, la totalidad del sistema de prueba.

Se pueden cumplir los requisitos de ejecución antes indicados en una implementación de un sistema de prueba estrechamente integrado. En tal caso, la totalidad del sistema de prueba TTCN-3 se implementa en una sola ejecución o proceso en el marco del cual se le asigna a cada entidad del sistema de prueba por lo menos una parte de la ejecución. En ese contexto, las operaciones TRI pueden implementarse como llamadas a procedimientos.

Ahora bien, los requisitos de ejecución también podrían cumplirse en una implementación del sistema de prueba TTCN-3 con una integración más flexible, por ejemplo con múltiples adaptadores SUT en un entorno de computación especializado. En este caso sólo una pequeña parte del adaptador SUT está estrechamente integrado con el resto del sistema de prueba TTCN-3, y los adaptadores SA reales pueden implementarse en procesos separados. Esa pequeña parte de SA podría entonces realizar solamente el encaminamiento de la información proporcionada por las operaciones TRI a los procesos del adaptador SUT deseados, los cuales posiblemente se ejecutarían en un computador distante, y viceversa.

## 5 Operaciones e interfaz de ejecución TTCN-3

En esta cláusula se estipula cuándo deben utilizarse las operaciones TRI y su efecto previsto en una implementación del sistema de prueba TTCN-3. Se define asimismo una serie de tipos de datos abstractos que luego se utiliza para la definición de las operaciones TRI. Esta definición incluye asimismo una descripción más detallada de los parámetros de entrada necesarios para cada llamada a una operación TRI y su valor de retorno.

### 5.1 Panorama general de la TRI

La TRI define la interacción entre las entidades TTCN-3 ejecutable (TE), adaptador SUT (SA) y adaptador de plataforma (PA), en una implementación de sistema de prueba TTCN-3. En términos conceptuales, la TRI proporciona un medio para que la TE envíe datos de prueba al SUT o manipule los temporizadores y, de manera similar, notifique a la TE los finales de temporización y los datos de prueba recibidos.

Cabe considerar que la TRI consta de dos subinterfaces, una interfaz triCommunication y una interfaz triPlatform. La interfaz triCommunication dirige la comunicación de una TTCN-3 ETS con el SUT, que se implementa en el SA. La interfaz triPlatform representa un conjunto de operaciones, que adaptan una ETS a una plataforma de ejecución determinada.

Ambas interfaces son bidireccionales, de modo que las partes llamante y llamada residen en las entidades TE, SA y PA del sistema de prueba. En el cuadro 1 se ilustra con mayores detalles la relación llamante/llamado entre las respectivas entidades. Obsérvese que en este cuadro se muestran únicamente las interacciones visibles en la TRI. La comunicación interna entre las partes de la misma entidad no queda reflejada, puesto que la estructura interna del TE, el SA o el PA puede ser diferente en una implementación de sistema de prueba TTCN-3.

**Cuadro 1/Z.144 – Panorama general de la interfaz**

| **Interfaz** | **Sentido (entidad llamante → entidad llamada)** | |
|---|---|---|
| Nombre | TE → SA o PA | SA o PA → TE |
| triCommunication | TE → SA | SA → TE |
| triPlatform | TE → PA | PA → TE |

### 5.1.1 Interfaz triCommunication

Esta interfaz consiste en las operaciones que son necesarias para la comunicación de la TTCN-3 ETS con el SUT. Esto incluye las operaciones destinadas a inicializar la interfaz del sistema de prueba (TSI), establecer las conexiones con el SUT y manipular las comunicaciones con esta última entidad basadas en mensajes y procedimientos. Además, la interfaz triCommunication ofrece una operación para reinicializar el adaptador SUT (SA).

### 5.1.2 Interfaz triPlatform

Esta interfaz incluye todas las operaciones necesarias para adaptar la TTCN-3 ejecutable a una plataforma de ejecución determinada. Esta interfaz ofrece medios para poner en marcha, parar o leer un temporizador, consultar su estado y añadir eventos de expiración a la lista de temporizadores expirados. Además, ofrece operaciones para llamar a las funciones externas TTCN-3 y reinicializar el adaptador de plataforma (PA). Cabe señalar que en la interfaz triPlatform no hay diferencia alguna entre los temporizadores explícitos e implícitos, sino que se recurrirá a cada uno de ellos uniformemente con su identificador de temporizador (Timer IDentifier, TID).

### 5.1.3 Correlación entre las llamadas a operaciones TTCN-3 y TRI

Algunas llamadas a una operación TTCN-3 tienen una correlación directa con una llamada a una operación TRI (o posiblemente dos en el caso de las operaciones TTCN-3 `execute` y `call`), tal como se muestra en el cuadro 2. Para todas las otras llamadas a operaciones TRI podría no haber correlación directa.

La correlación indicada para las operaciones de comunicación TTCN-3 (es decir, `send`, `call`, `reply` y `raise`) se mantiene únicamente si esas operaciones se llaman en un puerto de componente de prueba, que se hace corresponder a un puerto TSI. No obstante, esa correlación se mantiene para todos esos tipos de llamadas a operación si ningún componente del sistema ha sido especificado para un caso de prueba, es decir que para un caso de prueba se crea únicamente el componente de prueba MTC y ningún otro componente de prueba.

**Cuadro 2/Z.144 – Correlación entre las llamadas a las operaciones TTCN-3 y TRI (* = si es aplicable)**

| Nombre de la operación TTCN-3 | Nombre de la operación TRI | Nombre de la interfaz TRI |
|---|---|---|
| execute | triExecuteTestCase | TriCommunication |
| | triStartTimer* | TriPlatform |
| map | triMap | TriCommunication |
| unmap | triUnmap | TriCommunication |
| send | triSend (véase nota 1) | TriCommunication |
| | triSendBC (véase nota 2) | |
| | triSendMC (véase nota 3) | |
| call | triCall (véase nota 1) | TriCommunication |
| | triCallBC (véase nota 2) | |
| | triCallMC (véase nota 3) | |
| | triStartTimer* | TriPlatform |
| reply | triReply (véase nota 1) | TriCommunication |
| | triReply (véase nota 2) | |
| | triReply (véase nota 3) | |
| raise | triRaise (véase nota 1) | TriCommunication |
| | triRaise (véase nota 2) | |
| | triRaise (véase nota 3) | |
| action | triSUTactionInformal | TriCommunication |
| start (timer) | triStartTimer | TriPlatform |
| stop (timer) | triStopTimer | TriPlatform |
| read (timer) | triReadTimer | TriPlatform |
| running (timer) | triTimerRunning | TriPlatform |
| TTCN-3 external function | triExternalFunction | TriPlatform |
| NOTA 1 – Para la comunicación unidifusión.<br>NOTA 2 – Para la comunicación difusión.<br>NOTA 3 – Para la comunicación multidifusión. | | |

Obsérvese que la TE utiliza todas las operaciones TRI indicadas en el cuadro 2 y que puede implementar la llamada a estas operaciones de manera diferente cuando evalúe la situación instantánea de la TTCN dentro de la TTCN-3 ETS.

## 5.2 Gestión de errores

La gestión de errores explícita se especifica únicamente para las operaciones TRI llamadas por la TTCN-3 ejecutable (TE). El SA o PA informa sobre el estado de la operación TRI en el valor que devuelve una operación TRI. El valor de estado puede indicar el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación TRI. Por consiguiente, la TE puede reaccionar ante un error que se ha producido dentro del SA o el PA y generar, por ejemplo, un error de caso de prueba.

Para operaciones TRI que han sido llamadas por el SA o PA no se exige la gestión de errores explícita dado que esas operaciones se realizan en la TE. En este caso, la TE controla la ejecución de la prueba en el caso de que el error se produzca en una operación TRI de este tipo.

Obsérvese que los códigos de error específicos así como la detección y gestión de errores en cualquiera de las entidades del sistema de prueba están fuera del alcance de la presente especificación TRI.

## 5.3 Interfaz de datos

En las operaciones TRI sólo se pasarán como argumento datos de prueba codificados. La TTCN-3 ejecutable (TE) se encarga de codificar los datos de prueba que se enviarán y de codificar los datos de prueba recibidos en las correspondientes operaciones TRI, dado que las reglas de codificación pueden especificarse para un módulo TTCN-3 o dentro de éste. Obsérvese que la TE tiene que codificar los datos de prueba aun cuando no se haya facilitado información de codificación en una TTCN-3 ATS. En este caso, el fabricante de la herramienta tiene que definir una codificación.

En lugar de definir una interfaz de datos explícita para los tipos de datos TTCN-3 y ASN.1, la norma TRI define un conjunto de tipos de datos abstractos. Este tipo de datos se utiliza en la siguiente definición de operaciones TRI para indicar qué información ha de pasar como argumento la entidad llamante a la llamada y viceversa. La representación concreta de estos tipos de datos abstractos así como la definición de tipos de datos básicos se definen en la correspondiente correspondencia lingüística en las cláusulas 6 y 7.

Obsérvese que los valores de cualquier tipo de datos identificador deben ser únicos en la implementación del sistema de prueba, donde por únicos se entiende que sean globalmente distintos en cualquier instante.

Los siguientes tipos de datos abstractos se definen y utilizan para la definición de operaciones TRI.

### 5.3.1 Conexión

`TriComponentIdType` — El valor de tipo `TriComponentIdType` incluye un identificador, un nombre y el tipo de componente. El valor distinto de este último es el nombre del tipo del tipo de componente especificado en TTCN-3 ATS. Este tipo abstracto se utiliza principalmente para realizar operaciones de comunicación TRI en puertos TSI que están vinculados con muchos puertos de componente de prueba.

`TriComponentIdListType` — El valor de tipo `TriComponentIdListType` es una lista de `TriComponentIdType`. Este tipo abstracto se utiliza para comunicación multidifusión en TCI.

`TriPortIdType` — El valor de tipo `TriPortIdType` incluye un valor del tipo `TriComponentIdType` para representar el componente al que pertenece el puerto, el índice de puerto (si lo hubiera), y el nombre del puerto especificado en la TTCN-3 ATS. El tipo `TriPortIdType` se necesita principalmente para pasar información sobre la TSI y las conexiones con la TSI desde la TE hacia el SA.

`TriPortIdListType` — El valor de tipo `TriPortIdListType` es una lista de `TriPortIdType`. El tipo abstracto se utiliza para la inicialización después de llamar a un caso de prueba TTCN-3.

### 5.3.2 Comunicación

`TriMessageType` — El valor de tipo `TriMessageType` consiste en datos de prueba codificados que han de enviarse al SUT o se han recibido del SUT.

`TriAddressType` — El valor de tipo `TriAddressType` indica la dirección de origen o de destino dentro del SUT. Este tipo abstracto puede utilizarse en operaciones de comunicación TRI y es un tipo abierto, que es opaco a la TE.

`TriAddressListType` — El valor de tipo `TriAddressListType` es una lista de `TriAddressType`. Este tipo abstracto utiliza para la comunicación multidifusión en TRI.

`TriSignatureIdType` — El valor de tipo `TriSignatureIdType` es el nombre de una signatura de procedimiento especificada en TTCN-3 ATS. Este tipo abstracto se utiliza en operaciones de comunicación TRI basadas en procedimientos.

`TriParameterType` — El valor de tipo `TriParameterType` incluye un parámetro codificado y un valor de `TriParameterPassingModeType` para representar el modo de transmisión de argumentos especificado para el parámetro en la TTCN-3 ATS.

`TriParameterPassingModeType` — El valor de tipo `TriParameterPassingModeType` es *in*, *inout* o *out*. Este tipo abstracto se utiliza en operaciones de comunicaciones TRI basadas en procedimiento y para llamadas a funciones externas.

`TriParameterListType` — El valor de tipo `TriParameterListType` es una lista de `TriParameterType`. Este tipo abstracto se utiliza en las operaciones de comunicaciones TRI basadas en procedimiento y para llamadas a funciones externas.

`TriExceptionType` — El valor de tipo `TriExceptionType` es un tipo codificado y un valor de una excepción que ha de enviarse al SUT o se ha recibido de un SUT. Este tipo abstracto se utiliza en operaciones de comunicación TRI basadas en procedimiento.

### 5.3.3 Temporizador

`TriTimerIdType` — El valor del tipo `TriTimerIdType` especifica un identificador de un temporizador. Este tipo abstracto es obligatorio para todas las operaciones de temporizador TRI.

`TriTimerDurationType` — El valor de tipo `TriTimerDurationType` especifica la duración para un temporizador, en segundos.

### 5.3.4 Otros

`TriTestCaseIdType` — El valor del tipo `TriTestCaseIdType` es el nombre de un caso de prueba especificado en la TTCN-3 ATS.

`TriFunctionIdType` — El valor del tipo `TriFunctionIdType` es el nombre de una función externa especificado en la TTCN-3 ATS.

`TriStatusType` — El valor del tipo `TriStatusType` es ***TRI_OK*** o ***TRI_Error***, lo que indica el éxito o fallo de la operación TRI.

## 5.4 Descripción de operaciones

Todas las operaciones se definen utilizando el lenguaje de definición de interfaz (IDL). La correspondencia con el lenguaje concreto se define en las cláusulas 6 y 7.

En cada llamada a una operación TRI son obligatorios todos los parámetros *in*, *inout* y *out*, indicados en la definición de la operación de que se trate. El valor de los parámetros que acepta (*in*) los especifica la entidad llamante. Análogamente, el valor de los parámetros que devuelve (*out*) los especifica la entidad llamada. En el caso de los parámetros que acepta y devuelve (*inout*), el valor es especificado en primer lugar por la entidad llamante, pero la entidad llamada puede sustituirlo por uno nuevo. Obsérvese que aunque la TTCN-3 también utilice *in*, *inout* y *out* para las definiciones de signatura, las denotaciones utilizadas en la especificación TRI IDL no guardan relación con las de una especificación TTCN-3.

Las llamadas a operaciones deben utilizar un valor reservado para indicar la ausencia de parámetros que son opcionales en la correspondiente descripción de parámetros TRI. Los valores reservados para estos tipos se definen en el correspondiente lenguaje y se hará referencia posteriormente como al valor `null`.

Todas las funciones en la interfaz se definen utilizando la siguiente plantilla:

| **F.n.m** | **Nombre de operación** entidad llamante → entidad llamada |
|---|---|
| **Signatura** | Signatura IDL |
| **Parámetros que acepta** | Descripción de los datos que se pasan como parámetros para la operación, desde la entidad llamante hasta la entidad llamada |
| **Parámetros que devuelve** | Descripción de los datos que se pasan como parámetros para la operación, desde la entidad llamada hasta la entidad llamante |
| **Parámetros que acepta y devuelve** | Descripción de los datos que se pasan como parámetros para la operación, desde la entidad llamante hasta la entidad llamada y desde la entidad llamada hasta la entidad llamante |
| **Valor que devuelve** | Descripción del valor que devuelve la operación a la entidad llamante. |
| **Restricciones** | Descripción de las restricciones que se aplican a la llamada de la operación |
| **Efecto** | Función que debe realizar la entidad llamada antes de terminar la operación |

## 5.5 Operaciones de interfaz de comunicaciones

### 5.5.1 triSAReset (TE → SA)

| **Signatura** | `TriStatusType triSAReset()` |
|---|---|
| **Parámetros que acepta** | n.a. |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triSAReset`. El estado indica el éxito (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE puede llamar en cualquier momento a esta operación para reinicializar el SA |
| **Efecto** | El SA reinicializa todas las comunicaciones que mantiene activas, por ejemplo reinicializa las conexiones estáticas con el STU, cierra las conexiones dinámicas con el STU, descarta los mensajes o las llamadas de procedimientos pendientes. La operación `triResetSA` devuelve ***TRI_OK*** en caso de que la operación se haya realizado satisfactoriamente y ***TRI_Error*** en caso contrario. |

### 5.5.2 Operaciones de gestión de conexión

#### 5.5.2.1 triExecuteTestCase (TE → SA)

| **Signatura** | `TriStatusType triExecuteTestCase( in TriTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)` |
|---|---|
| **Parámetros que acepta** | `testCaseId` identificador del caso de prueba que va a ejecutarse<br>`tsiPortList` lista de los puertos de la interfaz del sistema de prueba definidos por el sistema de prueba |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triExecuteTestCase`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error*****)** de la operación. |
| **Restricciones** | La TE llama a esta operación inmediatamente después de ejecutar cualquier caso de prueba. El caso de prueba que va a ejecutarse se indica mediante `testCaseId`. `tsiPortList` contiene todos los puertos que se han declarado en la definición del componente del sistema para el caso de prueba, es decir, los puertos TSI. Si un componente de sistema no se ha definido explícitamente para caso de prueba en la TTCN-3 ATS, la `tsiPortList` contiene todos los puertos de comunicaciones del componente de prueba MTC. Los puertos en la `tsiPortList` figuran en el mismo orden en el que aparecen en la correspondiente declaración del componente TTCN-3. |
| **Efecto** | El SA puede establecer conexiones estáticas con el STU e inicializar mecanismos de comunicación con los puertos TSI.<br>La operación `triExecuteTestCase` devuelve ***TRI_OK*** en caso de que la operación se haya realizado con éxito y ***TRI_Error*** en caso contrario. |

#### 5.5.2.2 triMap (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triMap( in TriPortIdType compPortId, in TriPortIdType tsiPortId)` |
| **Parámetros que acepta** | `compPortId` identificador del puerto de componente de pruebas que ha de hacerse corresponder<br>`tsiPortId` identificador del puerto de la interfaz del sistema de pruebas que ha de hacerse corresponder |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triMap`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de establecimiento de correspondencia TTCN-3. |
| **Efecto** | El SA puede establecer una conexión dinámica con el STU para el puerto TSI de referencia.<br>La operación `triMap` devuelve ***TRI_Error*** en caso de que no haya sido posible establecer satisfactoriamente la conexión y ***TRI_OK*** en caso contrario. La operación debe devolver ***TRI_OK*** en caso de que no sea necesario que el sistema de pruebas establezca una conexión dinámica. |

#### 5.5.2.3 triUnmap (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triUnmap( in TriPortIdType compPortId, in TriPortIdType tsiPortId)` |
| **Parámetros que acepta** | `compPortId` identificador del puerto de componente de prueba para el que ha de deshacerse la correspondencia<br>`tsiPortId` identificador del puerto de interfaz de sistema de prueba para el que ha de deshacerse la correspondencia |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triUnmap`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación para deshacer una correspondencia de TTCN-3. |
| **Efecto** | El SA cerrará una conexión dinámica con el STU para el puerto TSI del caso. La operación `triMap` devuelve ***TRI_Error*** en caso de que no se haya podido establecer satisfactoriamente la conexión o que dicha operación no se hubiese establecido previamente y ***TRI_OK*** en caso contrario. La operación debe devolver ***TRI_OK*** en caso de que el sistema de pruebas no haya establecido conexiones dinámicas. |

### 5.5.3 Operaciones de comunicación mediante mensajes

#### 5.5.3.1 triSend (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triSend( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriMessageType sendMessage)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba transmisor<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía el mensaje al adaptador SUT<br>`SUTaddress` (opcional) dirección de destino dentro del SUT<br>`sendMessage` mensaje codificado que se va a enviar |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triSend`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de transmisión unidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de transmisión TTCN-3 si no se ha especificado un componente del sistema para el caso de prueba, es decir, se crea un solo componente de prueba MTC para cada caso de prueba.<br>La codificación de `sendMessage` se ha de realizar en la TE antes de llamar a la operación TRI. |
| **Efecto** | El SA envía el mensaje al SUT.<br>La operación `triSend` devuelve ***TRI_OK*** cuando se haya llevado a cabo satisfactoriamente. De lo contrario devolverá ***TRI_Error***. Obsérvese que el valor ***TRI_OK*** devuelto no implica que el SUT haya recibido el `sendMessage`. |

### 5.5.3.2 triSendBC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triSendBC( in TriComponentIdType componentId,`<br>`in TriPortIdType tsiPortId,`<br>`in TriMessageType sendMessage)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba transmisor<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual se envía el mensaje al adaptador SUT<br>`sendMessage` el mensaje codificado que va a enviarse |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triSend`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de transmisión por difusión TTCN-3 por un puerto de un componente, para el que se ha creado una correspondencia con un puerto TSI. La TE llama a esta operación para todas las operaciones de transmisión TTCN-3 si no se ha especificado un componente del sistema para el caso de prueba, es decir, se crea un solo componente de prueba MTC para cada caso de prueba.<br>La codificación de `sendMessage` ha de realizarse en la TE antes de realizar esta llamada a la operación TRI. |
| **Efecto** | El SA transmite por difusión el mensaje al SUT.<br>La operación `triSend` devuelve ***TRI_OK*** en caso de que se haya llevado a cabo satisfactoriamente. De lo contrario, devuelve ***TRI_Error***. Obsérvese que el valor ***TRI_OK*** no implica que el SUT haya recibido el `sendMessage`. |

### 5.5.3.3 triSendMC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triSendMC( in TriComponentIdType componentId,`<br>`in TriPortIdType tsiPortId,`<br>`in TriAddressListType SUTaddresses,`<br>`in TriMessageType sendMessage)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba transmisor<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual se envía el mensaje al adaptador SUT<br>`SUTaddresses` direcciones de destino dentro del SUT<br>`sendMessage` mensaje codificado que va a enviarse |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triSend`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de transmisión multidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de transmisión TTCN-3 si no se ha especificado un componente del sistema para el caso de prueba, es decir, se crea un solo componente de prueba MTC para cada caso de prueba.<br>La codificación de `sendMessage` tiene que realizarse en el TE antes de llamar a la operación TRI. |
| **Efecto** | El SA transmite por multidifusión el mensaje al SUT.<br>La operación `triSend` devuelve ***TRI_OK*** en caso de que se haya llevado a cabo satisfactoriamente. En caso contrario, devuelve ***TRI_Error***. Obsérvese que el valor ***TRI_OK*** devuelto no implica que el SUT haya recibido el `sendMessage`. |

#### 5.5.3.4 triEnqueueMsg (SA → TE)

| | |
|---|---|
| **Signatura** | `void triEnqueueMsg( in TriPortIdType tsiPortId,`<br>`in TriAddressType SUTaddress,`<br>`in TriComponentIdType componentId,`<br>`in TriMessageType receivedMessage)` |
| **Parámetros que acepta** | `tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual el adaptador SUT pone en cola del mensaje<br>`SUTaddress` (opcional) dirección de origen dentro del SUT<br>`componentId` identificador del componente de prueba receptor<br>`receivedMessage` mensaje recibido codificado |
| **Parámetros que devuelve** | n.a. |
| **Valor de retorno** | Ninguno |
| **Restricciones** | El SA llama a esta operación después de que haya recibido un mensaje procedente del SUT. Sólo puede utilizarse en caso de que el tsiPortId se haya hecho corresponder anteriormente con un puerto del componentId o que se haya hecho referencia al mismo en una declaración triExecuteTestCase anterior.<br>La llamada a la operación `triEnqueueMsg` debe contener un `receivedMessage` codificado. |
| **Efecto** | Esta operación pasará el mensaje a la TE indicando el componentId del componente al que existe la correspondencia con el tsiPortId del puerto TSI.<br>La decodificación del receivedMessage se debe realizar en la TE. |

### 5.5.4 Operaciones de comunicación mediante procedimientos

#### 5.5.4.1 triCall (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triCall( in TriComponentIdType componentId,`<br>`in TriPortIdType tsiPortId,`<br>`in TriAddressType SUTaddress,`<br>`in TriSignatureIdType signatureId,`<br>`in TriParameterListType parameterList)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba que realiza la llamada al procedimiento<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual se envía la llamada al procedimiento hacia el adaptador SUT<br>`SUTaddress` (opcional) direcciones de destino dentro del SUT<br>`signatureId` identificador de la signatura de la llamada de procedimiento<br>`parameterList` lista de parámetros codificados que forman parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3 |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve la situación de la operación `triCall`. La situación indica éxito local (***TRI_OK***) o fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de llamada unidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de llamada TTCN-3 si no se ha especificado el componente del sistema para un caso de prueba, es decir, se crea un solo componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros de procedimiento *in* e *inout* contienen valores codificados.<br>Los parámetros del procedimiento son los parámetros especificados en la plantilla de signatura TTCN-3. Su codificación se ha de realizar en la TE antes de llamar a la operación TRI. |
| **Efecto** | Al llamar a esta operación, el SA puede iniciar la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triCall` devuelve sin esperar que termine la llamada al procedimiento realizada (véase la nota). Esta operación TRI devuelve *TRI_OK* si la llamada al procedimiento se ha llevado a cabo con éxito, y *TRI_Error* en caso contrario. El SA no debe indicar error en caso de que el valor de alguno de los parámetros que devuelve sea no nulo. Obsérvese que el Valor que devuelve de esta operación TRI no guarda relación alguna con el éxito o fallo de la llamada al procedimiento.<br>Obsérvese que en la signatura de la operación `triCall` *no* se incluye un valor de temporización opcional, que puede especificarse en la TTCN-3 ATS para una operación de llamada. La TE es responsable de gestionar este asunto mediante la puesta en marcha de un temporizador para la operación de llamada TTCN-3 en el PA con una llamada de operación TRI diferente, es decir, `triStartTimer`. |
| NOTA – Esto podría lograrse por ejemplo generando un nuevo proceso o tarea. No obstante, la gestión de esta llamada procedimiento depende de la implementación de la TE. | |

### 5.5.4.2 triCallBC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triCallBC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriSignatureIdType signatureId, in TriParameterListType parameterList)` |
| **Parámetros que acepta** | `componenteId` identificador del componente de prueba que realiza la llamada al procedimiento<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía la llamada al procedimiento hacia el adaptador SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento<br>`parameterList` lista de parámetros codificados que forman parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3 |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triCall`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de llamada de difusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de llamada TTCN-3 si no se ha especificado un componente de sistema para el caso de prueba, es decir, sólo se crea un componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros del procedimiento *in* e *inout* contienen valores codificados.<br>Los parámetros del procedimiento son los parámetros especificados en la plantilla de signatura TTCN-3. Su codificación tiene que realizarse en el TE antes de llamar a esta operación TRI. |
| **Efecto** | Al llamar a esta operación el SA puede iniciar y transmitir por difusión la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y de puerto TSI `tsiPortId`.<br>La operación `triCall` debe devolver el valor sin esperar a que termine la llamada al procedimiento realizada (véase la nota). Esta operación TRI devuelve *TRI_OK* si la llamada al procedimiento se inició correctamente, y *TRI_Error* en caso contrario. El SA no debe implicar error alguno en caso de que el valor de cualquier parámetro que devuelve sea no nulo. Obsérvese que el valor que devuelve esta operación TRI es independiente del éxito o el fallo de la llamada al procedimiento.<br>Obsérvese asimismo que el valor de temporizador opcional, que puede especificarse en la TTCN-3 ATS para la operación de llamada, *no* incluye la signatura de operación `triCall`. La TE es responsable de resolver este asunto mediante la puesta en marcha de un temporizador para la operación de llamada TTCN-3 en el PA con una llamada de operación TRI independiente, es decir `triStartTimer`. |
| NOTA – Esto puede lograrse mediante, por ejemplo, la generación de un nuevo proceso o tarea. No obstante, la gestión de esta llamada de procedimiento depende de la implementación de la TE. | |

### 5.5.4.3 triCallMC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triCallMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriParameterListType parameterList)` |
| **Parámetros que acepta** | `componenteId` identificador del componente de prueba que realiza la llamada al procedimiento<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía la llamada al procedimiento hacia el adaptador SUT<br>`SUTaddresses` direcciones de destino dentro de la SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento<br>`parameterList` lista de parámetros codificados que forman parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3 |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triCall`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de llamada multidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de llamada TTCN-3 si no se ha especificado un componente de sistema para el caso de prueba, es decir, sólo se crea un componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros del procedimiento *in* e *inout* contienen valores codificados.<br>Los parámetros del procedimiento son los parámetros especificados en la plantilla de signaturas TTCN-3. Su codificación tiene que realizarse en la TE antes de llamar a esta operación TRI. |
| **Efecto** | Al llamar a esta operación el SA puede iniciar y transmitir por difusión la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triCall` debe devolver el valor sin esperar a que termine la llamada al procedimiento realizada (véase la nota). Esta operación TRI devuelve *TRI_OK* si la llamada al procedimiento se inició correctamente, y *TRI_Error* en caso contrario. El SA no debe indicar error alguno en caso de que el valor de cualquier parámetro que devuelve sea no nulo. Obsérvese que el valor que devuelve esta operación TRI es independiente del éxito o el fallo de la llamada al procedimiento.<br>Obsérvese asimismo que el valor de temporizador opcional, que puede especificarse en la TTCN-3 ATS para la operación de llamada, no incluye la signatura de operación `triCall`. La TE es responsable de resolver este problema mediante la puesta en marcha de un temporizador para la operación de llamada TTCN-3 en el PA con una llamada de operación TRI independiente, es decir `triStartTimer`. |
| NOTA – Esto puede lograrse mediante, por ejemplo, la generación de un nuevo proceso o tarea. No obstante, la gestión de esta llamada al procedimiento depende de la implementación de la TE. | |

#### 5.5.4.4 triReply (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triReply( in TriComponentIdType componentId,`<br>`in TriPortIdType tsiPortId,`<br>`in TriAddressType SUTaddress,`<br>`in TriSignatureIdType signatureId,`<br>`in TriParameterListType parameterList,`<br>`in TriParameterType returnValue)` |
| **Parámetros que acepta** | `componenteId` identificador del componente de prueba que responde<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual se envía la respuesta hacia el adaptador SUT<br>`SUTaddress` (opcional) direcciones de destino dentro del SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento<br>`parameterList` lista de parámetros codificados que forma parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3<br>`returnValue` (opcional) valor codificado que devuelve la llamada al procedimiento |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triReply`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de respuesta unidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de respuesta TTCN-3 si no se ha especificado ninguna componente de sistema para el caso de prueba, es decir, si únicamente se crea un componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros del procedimiento *out* e *inout* y el valor que devuelven contienen valores codificados.<br>La `parameterList` contiene parámetros de llamada al procedimiento. Esos parámetros son los especificados en la plantilla de signatura TTCN-3. Su decodificación tiene que realizarse en la TE antes de realizar la llamada de operación TRI.<br>Si no se ha definido ningún tipo de retorno para la forma del procedimiento en la TTCN-3 ATS, se devolverá un valor distinto de `null`. |
| **Efecto** | Al llamar a esta operación el SA puede responder a la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triReply` devolverá ***TRI_OK*** si se ha llevado a cabo con éxito y ***TRI_Error*** en caso contrario. El SA no indicará error en caso de que el valor de cualquier parámetro que acepta o el valor indefinido que devuelva sea diferente de `null`. |

### 5.5.4.5 triReplyBC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triReplyBC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriSignatureIdType signatureId, in TriParameterListType parameterList, in TriParameterType returnValue)` |
| **Parámetros que acepta** | `componenteId` identificador del componente de prueba que responde<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual se envía la respuesta hacia el adaptador SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento<br>`parameterList` lista de parámetros codificados que forma parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3<br>`returnValue` (opcional) valor codificado que devuelve la llamada al procedimiento |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triReply`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de respuesta difusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de respuesta TTCN-3 si no se ha especificado ninguna componente de sistema para el caso de prueba, es decir, únicamente se crea un componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros del procedimiento *out* e *inout* y el valor que devuelven contienen valores codificados. La `parameterList` contiene parámetros de llamada al procedimiento. Esos parámetros son los especificados en la plantilla de signatura TTCN-3. Su decodificación tiene que realizarse en la TE antes de realizar la llamada de operación TRI.<br>Si no se ha definido ningún tipo de retorno para la forma del procedimiento en la TTCN-3 ATS, se devolverá un valor distinto de `null`. |
| **Efecto** | Al llamar a esta operación el SA puede responder a la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triReply` devolverá ***TRI_OK*** si se ha llevado a cabo con éxito, y ***TRI_Error*** en caso contrario. El SA no indicará error en caso de que el valor de cualquier parámetro que devuelve o el valor indefinido que devuelva sea diferente de `null`. |

### 5.5.4.6 triReplyMC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triReplyMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriParameterListType parameterList, in TriParameterType returnValue)` |
| **Parámetros que acepta** | `componenteId` identificador del componente de prueba que responde<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía la respuesta hacia el adaptador SUT<br>`SUTaddresses` (opcional) direcciones de destino dentro del SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento<br>`parameterList` lista de parámetros codificados que forma parte de la signatura indicada. Los parámetros en `parameterList` figuran en el mismo orden en el que aparecen en la declaración de signatura TTCN-3<br>`returnValue` (opcional) valor codificado que devuelve la llamada al procedimiento |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triReply`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de respuesta multidifusión TTCN-3 por un puerto de un componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones de respuesta TTCN-3 si no se ha especificado ninguna componente de sistema para el caso de prueba, es decir, únicamente se crea un componente de prueba MTC para cada caso de prueba.<br>Todos los parámetros del procedimiento *out* e *inout* y el valor que devuelven contienen valores codificados. La `parameterList` contiene parámetros de llamada al procedimiento. Esos parámetros son los especificados en la plantilla de signatura TTCN-3. Su decodificación tiene que realizarse en la TE antes de realizar la llamada de operación TRI.<br>Si no se ha definido ningún tipo de retorno para la forma del procedimiento en la TTCN-3 ATS, se devolverá un valor distinto de `null`. |
| **Efecto** | Al llamar a esta operación el SA puede responder a la llamada al procedimiento correspondiente al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triReply` devolverá ***TRI_OK*** si se ha llevado a cabo con éxito, y ***TRI_Error*** en caso contrario. El SA no indicará error en el caso de que el valor de cualquier parámetro que devuelve o el valor indefinido que devuelva sea diferente de `null`. |

### 5.5.4.7 triRaise (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triRaise( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriSignatureIdType signatureId, in TriExceptionType exc)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba que genera la excepción<br>`tsiPortId` identificador del puerto de la interfaz de sistema de prueba por el que se envía la excepción hacia el adaptador SUT<br>`SUTaddress` (opcional) direcciones de destino dentro del SUT<br>`signatureId` identificador de la signatura de la llamada al procedimiento con el que está relacionada la excepción<br>`exc` la excepción codificada |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triRaise`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación generar unidifusión TTCN-3 en un puerto del componente, que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones general TTCN-3 si no se ha especificado en ningún componente de sistema para el caso de prueba, es decir, sólo se crea un componente de prueba MTC para cada caso de prueba.<br>La codificación de la excepción se tiene que realizar en la TE antes de llamar a la operación TRI. |
| **Efectos** | Al llamar a esta operación el SA puede genera una excepción hacia una llamada al procedimiento correspondiente con el identificador de signatura `signatureId` y el puerto TSI `tsiPortId`.<br>La operación `triRaise` devuelve *TRI_OK* si se ha llevado a cabo con éxito y *TRI_Error* en caso contrario. |

### 5.5.4.8 triRaiseBC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triRaiseBC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriSignatureIdType signatureId, in TriExceptionType exc)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba que genera la excepción<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía la excepción hacia el adaptador SUT<br>`signatureId` identificador de la signatura de prueba de la llamada al procedimiento que está relacionado con la excepción<br>`exc` la excepción codificada |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triRaise`. Este estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación generar radiodifusión TTCN-3 en un puerto de componente que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones generadas TTCN-3 si no se ha especificado ningún componente del sistema para el caso de prueba, es decir, sólo se crea un componente de prueba MTC para cada caso de prueba.<br>La codificación de la excepción ha de realizarse en la TE antes de llamar a la operación TRI. |
| **Efecto** | Al invocar esta operación el SA puede generar y difundir una excepción a las llamadas de procedimiento correspondientes al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triRaise` devuelve *TRI_OK* si se ha completado con éxito y *TRI_Error* en caso contrario. |

### 5.5.4.9 triRaiseMC (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triRaiseMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriExceptionType exc)` |
| **Parámetros que acepta** | `componentId` identificador del componente de prueba que genera la excepción<br>`tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el que se envía la excepción hacia el adaptador SUT<br>`SUTaddresses` direcciones de destino dentro del SUT<br>`signatureId` identificador de la signatura de prueba de la llamada al procedimiento que está relacionado con la excepción<br>`exc` la excepción codificada |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triRaise`. Este estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación generar multidifusión TTCN-3 en un puerto de componente que se ha hecho corresponder con un puerto TSI. La TE llama a esta operación para todas las operaciones generadas TTCN-3 si no se ha especificado ningún componente del sistema para el caso de prueba, es decir, sólo se crea un componente de prueba MTC para cada caso de prueba.<br>La codificación de la excepción ha de realizarse en el TE antes de llamar a la operación TRI. |
| **Efecto** | Al invocar esta operación el SA puede generar y difundir una excepción a las llamadas de procedimiento correspondientes al identificador de signatura `signatureId` y al puerto TSI `tsiPortId`.<br>La operación `triRaise` devuelve *TRI_OK* si se ha completado con éxito y *TRI_Error* en caso contrario. |

### 5.5.4.10 triEnqueueCall (SA → TE)

| | | |
|---|---|---|
| **Signatura** | `void triEnqueueCall(  in TriPortIdType tsiPortId,`<br>`in TriAddressType SUTaddress,`<br>`in TriComponentIdType componentId,`<br>`in TriSignatureIdType signatureId,`<br>`in TriParameterListType parameterList)` | |
| **Parámetros que acepta** | `tsiPortId` | identificador del puerto de la interfaz del sistema de prueba por el cual el adaptador SUT pone en cola la llamada al procedimiento |
| | SUTaddress | (opcional) direcciones de origen dentro del SUT |
| | componentId | identificador del componente de prueba receptor |
| | `signatureId` | identificador de la signatura de la llamada al procedimiento |
| | `parameterList` | lista de parámetros codificados que forman parte de la signatura indicada. Los parámetros en `parameterList` están en el orden en el que aparecen en la declaración de signatura de TTCN-3. Descripción de los datos pasados como argumentos a la operación desde la entidad llamante a la entidad llamada |
| **Parámetros que devuelve** | n.a. | |
| **Valor que devuelve** | ninguno | |
| **Restricciones** | El SA puede llamar a esta operación después de que haya recibido una llamada al procedimiento desde el SUT. Sólo puede utilizarse cuando `tsiPortId` se haya hecho corresponder previamente con un puerto del `componentId` o se haya hecho referencia en una declaración `triExecuteTestCase` anterior.<br>Al invocar a una operación `triEnqueueCall` todos los parámetros de procedimiento *in* e *inout* contienen valores codificados. | |
| **Efecto** | La TE puede poner en cola esta llamada al procedimiento con el identificador de signatura `signatureId` en el puerto del componente `componentId` con el que se ha hecho corresponder el puerto TSI `tsiPortId`. La decodificación de los parámetros de procedimiento tienen que realizarse en la TE.<br>La TE no indicará error alguno en caso de que el valor de cualquiera de los parámetros *que devuelve* sea diferente de nulo. | |

### 5.5.4.11 triEnqueueReply (SA → TE)

| | | |
|---|---|---|
| **Signatura** | `void triEnqueueReply(  in TriPortIdType tsiPortId,`<br>`in TriAddressType SUTaddress,`<br>`in TriComponentIdType componentId,`<br>`in TriSignatureIdType signatureId,`<br>`in TriParameterListType parameterList,`<br>`in TriParameterType returnValue)` | |
| **Parámetros que acepta** | `tsiPortId` | identificador del puerto de la interfaz del sistema de prueba por el que se envía la excepción hacia el adaptador SUT |
| | `SUTaddress` | direcciones de destino dentro del SUT |
| | `componentId` | identificador del componente de prueba que genera la excepción |
| | `signatureId` | identificador de la signatura de la llamada al procedimiento |
| | `parameterList` | lista de parámetros codificados que forman parte de la signatura indicada. Los parámetros en `parameterList` están en el orden en el que aparecen en la declaración de signatura de TTCN-3. |
| | `returnValue` | (opcional) valor codificado que devuelve la llamada al procedimiento |
| **Parámetros que devuelve** | n.a. | |
| **Valor que devuelve** | ninguno | |
| **Restricciones** | El SA puede llamar a esta operación después de que haya recibido una respuesta procedente del SUT. Sólo puede utilizarse cuando `tsiPortId` se haya hecho corresponder previamente con un puerto de `componentId` o se haya hecho referencia en una declaración `triExecuteTestCase` anterior.<br>Al llamar una operación `triEnqueueReply` todos los parámetros de los procedimientos *out* e *inout* y el valor que se devuelve tienen que estar codificados.<br>Si no se ha definido un tipo de retorno para la signatura del procedimiento en TTCN-3 ATS, el valor que devuelve deberá ser el valor distintivo `null`. | |
| **Efecto** | La TE puede poner en cola esta respuesta a la llamada de procedimiento con el identificador de signatura `signatureId` en el puerto del componente `componentId` con el que el puerto TSI `tsiPortId` se ha hecho corresponder. La decodificación de los parámetros de procedimiento tiene que realizarse dentro de la TE.<br>La TE no indicará error alguno en caso de que el valor de cualquiera de los parámetros *que acepta* o el valor indefinido que devuelva sea diferente de null. | |

#### 5.5.4.12 triEnqueueException (SA → TE)

| | |
|---|---|
| **Signatura** | `void triEnqueueException( in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriComponentIdType componentId, in TriSignatureIdType signatureId, in TriExceptionType exc)` |
| **Parámetros que acepta** | `tsiPortId` identificador del puerto de la interfaz del sistema de prueba por el cual el adaptador SUT pone en cola la excepción<br>`SUTaddress` (opcional) dirección de origen dentro del SUT<br>`componentId` identifier of the receiving test component<br>`signatureId` identificador de la signatura de la llamada al procedimiento con el que está relacionada la excepción<br>`exc` la excepción codificada |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | ninguno |
| **Restricciones** | El SA puede llamar a esta operación después de que haya recibido una respuesta del SUT. Sólo puede utilizarse cuando `tsiPortId` se haya hecho corresponder anteriormente con un puerto de `componentId` o se haya hecho referencia en una declaración `triExecuteTestCase` anterior.<br>Al invocar una operación `triEnqueueException`, la `exception` debe estar codificada. |
| **Efecto** | La TE puede poner en cola esta excepción para la llamada al procedimiento con el identificador de signatura `signatureId` en el puerto del componente `componentId` con cual se ha hecho corresponder el puerto TSI `tsiPortId`.<br>La decodificación de la excepción tiene que realizarse dentro de la TE. |

### 5.5.5 Operaciones diversas

#### 5.5.5.1 triSUTactionInformal (TE → SA)

| | |
|---|---|
| **Signatura** | `TriStatusType triSUTactionInformal(in string description)` |
| **Parámetros que acepta** | `description` descripción informal de una acción que ha de realizar el SUT |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triSUTactionInformal`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando ejecuta una operación de acción TTCN-3 SUT, que sólo contiene una cadena. |
| **Efecto** | Tras llamar a esta operación el SA iniciará las acciones descritas que habrá de realizar el SUT, por ejemplo encendido, inicialización, o envío de un mensaje al SUT.<br>La operación `triSUTactionInformal` devuelve *TRI_OK* si se ha llevado a cabo con éxito y *TRI_Error* en caso contrario. Obsérvese que el valor que devuelve esta operación TRI no indica el éxito o el fallo de las acciones que realice el SUT. |

#### 5.5.5.2 triSUTactionTemplate (TE → SA)

Obsoleto.

## 5.6 Operaciones de interfaz de plataformas

### 5.6.1 triPAReset (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triPAReset()` |
| **Parámetros que acepta** | n.a. |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triPAReset`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE puede llamar a esta operación en cualquier momento para reinicializar el PA. |
| **Efecto** | El PA reinicializará todas las actividades de temporización que esté llevando a cabo en ese momento, por ejemplo detendrá todos los temporizadores que estén en marcha y descartará todas las temporizaciones pendientes de los temporizadores que hayan expirado.<br>La operación `triResetSA` devuelve *TRI_OK* si se ha realizado satisfactoriamente y *TRI_Error* en caso contrario. |

### 5.6.2 Operaciones de temporización

#### 5.6.2.1 triStartTimer (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triStartTimer( in TriTimerIdType timerId, in TriTimerDurationType timerDuration)` |
| **Parámetros que acepta** | `timerId` identificador del temporizador de que se trata<br>`timerDuration` duración del temporizador en segundos |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triStartTimer`. Este estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando necesita poner en marcha un temporizador |
| **Efecto** | Al llamar a esta operación el PA pondrá en marcha el temporizador indicado para la duración especificada. El temporizador cuenta desde el valor cero (0,0) hasta el valor máximo especificado por timerDuration. Si el temporizador especificado por `timerId` ya estuviese en marcha, se reinicializará. Tras la expiración del temporizador, el PA llamará a la operación `triTimeout()` pasándole como argumento `timerId`.<br>La operación `triStartTimer` devuelve *TRI_OK* si el temporizador se ha puesto en marcha correctamente, y *TRI_Error* en caso contrario. |

#### 5.6.2.2 triStopTimer (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triStopTimer( in TriTimerIdType timerId)` |
| **Parámetros que acepta** | `timerId` identificador del temporizador de que se trata |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | Devuelve el estado de la operación `triStopTimer`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación para detener un temporizador |
| **Efecto** | Al llamar a esta operación el PA utilizará el `timerId` para detener el temporizador especificado. Si se trata de detener un temporizador inactivo, es decir, un temporizador que no se ha puesto en marcha o que ya ha expirado, esta operación no tendrá efecto alguno.<br>La operación `triStopTimer` devuelve ***TRI_OK*** si la operación se ha realizado con éxito, y ***TRI_Error*** en caso contrario. Obsérvese que detener un temporizador inactivo es una operación válida. En este caso se devolverá ***TRI_OK***. |

#### 5.6.2.3 triReadTimer (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triReadTimer( in TriTimerIdType timerId, out TriTimerDurationType elapsedTime)` |
| **Parámetros que acepta** | `timerId` identificador del temporizador de que se trata |
| **Parámetros que devuelve** | `elapsedTime` valor del tiempo transcurrido desde que se puso en marcha el temporizador en segundos |
| **Valor que devuelve** | Devuelve el estado de la operación `triReadTimer`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando desea realizar la operación leer temporizador TTCN-3 para el temporizador indicado (véase 5.3.1). |
| **Efecto** | Al llamar a esta operación el PA utilizará el `timerId` para obtener el tiempo transcurrido desde que se puso en marcha el temporizador. El valor que devuelve `elapsedTime` se expresará en segundos. La lectura de un temporizador inactivo, es decir, un temporizador que no se ha puesto en marcha o que ya ha expirado, dará como resultado un valor de tiempo transcurrido igual a cero.<br>La operación `triReadTimer` devuelve ***TRI_OK*** si ésta se ha realizado correctamente, y ***TRI_Error*** en caso contrario. |

#### 5.6.2.4 triTimerRunning (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triTimerRunning( in TriTimerIdType timerId, out boolean running)` |
| **Parámetros que acepta** | `timerId` identificador del temporizador de que se trata |
| **Parámetros que devuelve** | `running` estado del temporizador |
| **Valor que devuelve** | Devuelve el estado de la operación `triTimerRunning`. El estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando quiere ejecutar la operación comprobar el temporizador en marcha TTCN-3 en el temporizador especificado (véase 5.3.1). |
| **Efecto** | Al llamar a esta operación, el PA utilizará el `timerId` para obtener el estado del temporizador. La operación asignará a `running` el valor booleano `true` si el temporizador está en marcha en ese momento.<br>La operación `triTimerRunning` devuelve ***TRI_OK*** si el estado del temporizador se ha determinado correctamente, y ***TRI_Error*** en caso contrario. |

#### 5.6.2.5 triTimeout (PA → TE)

| | |
|---|---|
| **Signatura** | `void triTimeout(in TriTimerIdType timerId)` |
| **Parámetros que acepta** | `timerId` identificador del temporizador de que se trata |
| **Parámetros que devuelve** | n.a. |
| **Valor que devuelve** | ninguno |
| **Restricciones** | El PA llama a esta operación después de que haya expirado, es decir, haya alcanzado el valor de la duración máxima, un temporizador que se había puesto en marcha utilizando la operación `triStartTimer`. |
| **Efecto** | La temporización del `timerId` puede añadirse a la lista de temporizaciones de la TE. La implementación de esa operación en la TE debe ser tal que permita gestionar las diferentes semánticas TTCN-3 para temporizadores definidas en la Rec. UIT-T Z.143 [4] (véase también 5.3.1). |

### 5.6.3 Otras operaciones

#### 5.6.3.1 triExternalFunction (TE → PA)

| | |
|---|---|
| **Signatura** | `TriStatusType triExternalFunction( in TriFunctionIdType functionId, inout TriParameterListType parameterList, out TriParameterType returnValue)` |
| **Parámetros que acepta** | `functionId` identificador de la función externa |
| **Parámetros que devuelve** | `returnValue` (opcional) valor codificado que devuelve |
| **Parámetros que acepta y devuelve** | `parameterList` una lista de los parámetros codificados para la función especificada. Los parámetros en `parameterList` figuran en el orden en el que aparecen en la declaración de la función TTCN-3. |
| **Valor que devuelve** | Devuelve el estado de la operación `triExternalFunction`. Este estado indica el éxito local (***TRI_OK***) o el fallo (***TRI_Error***) de la operación. |
| **Restricciones** | La TE llama a esta operación cuando desea ejecutar una función definida como externa TTCN-3 (es decir, no todas las funciones no externas están implementadas dentro de la TE).<br>Cuando la TE llama a una operación `triExternalFunction`, el valor de todos los parámetros que acepta (*in*) y que acepte y devuelve (*inout*) la función están codificados. El PA no debe señalar error en caso de que el valor de cualquier parámetro que devuelve (*out*) sea no nulo. |
| **Efecto** | El PA debe implementar el comportamiento de cada función externa especificada en la TTCN-3 ATS. Al llamar a esta operación, el PA llamará a la función indicada por el identificador `functionId`. Tendrá acceso a los parámetros que acepta y que acepte y devuelve esa función los cuales figuran en la `parameterList`, evaluará la función externa utilizando los valores de estos parámetros y calculará los valores para los parámetros que acepte y devuelve (*inout*) de la `parameterList`. La operación deberá devolver valores codificados de todos los parámetros que acepta y devuelve la función y el valor codificado que devuelve la función externa.<br>Si no se ha definido ningún tipo de retorno para el valor que devuelve esta función externa en la TTCN-3 ATS, para esta última se utiliza el valor distintivo `null`.<br>La operación `triExternalFunction` devuelve ***TRI_OK*** si el PA lleva a cabo la evaluación de la función externa correctamente, y ***TRI_Error*** en caso contrario.<br>Obsérvese que mientras todas las operaciones TRI no pueden bloquear el funcionamiento, la operación `triExternalFunction` sí puede *bloquearlo.* Esto significa que esta operación no termina hasta tanto la función externa indicada se haya ejecutado completamente. Las funciones externas se han de implementar meticulosamente dado que pueden detener indefinidamente la ejecución del componente de prueba e incluso de la implementación de todo el sistema de prueba. |

# 6 Correspondencia con el lenguaje Java

## 6.1 Introducción

En esta cláusula se describe la relación entre el lenguaje Java y la TRI. Por razones prácticas, se utiliza una correspondencia lingüística especializada en lugar de utilizar el OMG IDL para el lenguaje Java.

En la correspondencia con el lenguaje Java para la interfaz de ejecución TTCN-3 se definen las equivalencias entre las definiciones IDL descritas en la cláusula 5 y el lenguaje Java. Esta correspondencia lingüística es independiente de la versión de Java utilizada dada que sólo se utilizan estructuras básicas del lenguaje Java.

## 6.2 Nombres y alcances

### 6.2.1 Nombres

Aunque no existan discrepancias entre los identificadores utilizados en la definición IDL y el lenguaje Java es necesario aplicar algunas reglas de traducción de nombres a los identificadores IDL.

- Los identificadores de parámetros Java deben comenzar en minúscula y la otra parte integrante del identificador del parámetro debe comenzar en mayúscula. Por ejemplo el identificador del parámetro IDL **`SUTaddress`** se escribe `sutAddress` en Java.
- Los identificadores de clase o interfaces Java no contienen el sufijo `Type` utilizado en la definición IDL. Por ejemplo el tipo IDL **`TriPortIdType`** corresponde a `TriPortId` en Java.

La correspondencia resultante es conforme con los convenios de codificación Java normalizados.

### 6.2.2 Alcances

El módulo IDL **`triInterface`** figura en el lote Java `org.etsi.ttcn3.tri`. Todas las declaraciones de tipo IDL de este módulo se hacen corresponder con las declaraciones de interfaces o clases Java de este lote.

## 6.3 Correspondencia de tipos

### 6.3.1 Correspondencia de tipos básicos

En el cuadro 3 figura una descripción general de la correspondencia entre los tipos IDL básicos y los tipos Java.

**Cuadro 3/Z.144 – Correspondencia de tipos básicos**

| Tipo IDL | Tipo Java |
|---|---|
| `boolean` | `org.etsi.ttcn.tri.TriBoolean` |
| `string` | `java.lang.String` |

Los demás tipos básicos IDL no se utilizan dentro de la definición IDL.

#### 6.3.1.1 Booleano

El tipo **`boolean`** IDL corresponde con la interfaz `org.etsi.ttcn.tri.TriBoolean`, de modo que los objetos que integren esta interfaz pueden actuar como objetos contenedores.

Se define la siguiente interfaz para `org.etsi.ttcn.tri.TriBoolean`:

```
// TriBoolean
package org.etsi.ttcn.tri;
public interface TriBoolean {
    public void setBooleanValue(boolean value);
    public boolean getBooleanValue();
}
```

##### 6.3.1.1.1 Métodos

– `setBooleanValue(boolean value)`
Asigna a este `TriBoolean` el valor booleano `value`.

- `getBooleanValue()`
Devuelve el valor booleano representado por este `TriBoolean`.

#### 6.3.1.2 Cadena

El tipo **`string`** IDL corresponde a la clase `java.lang.String` sin verificación de gama ni limitación del número de caracteres en la cadena. Todas las posibles cadenas definidas en TTCN-3 pueden convertirse a `java.lang.String`.

### 6.3.2 Correspondencia de tipos estructurados

En la descripción IDL de la TRI los tipos definidos por el usuario se consideran tipos nativos. En lenguaje Java, estos tipos corresponden a interfaces Java. Las interfaces definen métodos y atributos que pueden utilizar los objetos que integren esta interfaz.

#### 6.3.2.1 TriPortIdType

**`TriPortIdType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriPortIdType
package org.etsi.ttcn.tri;
public interface TriPortId {
    public String getPortName();
    public TriComponentId getComponent();
    public boolean isArray();
    public int getPortIndex();
}
```

##### 6.3.2.1.1 Métodos

- `getPortName()`
Devuelve el nombre del puerto definido en la especificación TTCN-3.

– `getComponent()`

Devuelve el identificador del componente al que pertenece este `TriPortId` definido en la especificación de TTCN-3.

– `isArray()`
Devuelve `true` si el puerto forma parte de un conjunto de puertos y `false` en caso contrario.

– `getPortIndex()`
Si este puerto forma de un conjunto de puertos, devuelve el índice puerto comenzando desde cero. Si el puerto no forma parte de un conjunto de puertos, devuelve –1.

#### 6.3.2.2 TriPortIdListType

**`TriPortIdListType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriPortIdListType
package org.etsi.ttcn.tri;
public interface TriPortIdList {
    public int size();
    public boolean isEmpty();
    public java.util.Enumeration getPortIds();
    public TriPortId get(int index);
}
```

##### 6.3.2.2.1 Métodos

`size()`
Devuelve el número de puertos que contiene esta lista.

`isEmpty()`
Devuelve `true` si la lista no contiene puertos.

`getPortIds()`
Devuelve una `Enumeration` de todos los puertos en la lista. Los puertos en esta enumeración figuran en el mismo orden en el que aparecen en la lista.

`get(int index)`
Devuelve el `TriPortId` en la posición especificada.

#### 6.3.2.3 TriComponentIdType

**`TriComponentIdType`** corresponde a la siguiente interfaz:

```
// TRI IDL TriComponentIdType
package org.etsi.ttcn.tri;
public interface TriComponentId {
    public String getComponentId();
    public String getComponentName();
    public String getComponentTypeName();
    public TriPortIdList getPortList();
    public boolean equals(TriComponentId port);
}
```

##### 6.3.2.3.1 Métodos

– `getComponentId()`
Devuelve una representación de este identificador único del componente.

– `getComponentName()`
Devuelve el nombre del componente definido en la especificación TTCN-3. Si el componente no tiene nombre, devuelve una cadena vacía.

– `getComponentTypeName()`
Devuelve el nombre del tipo de componente definido en la especificación TTCN-3.

– `getPortList()`
Devuelve la lista de puertos del componente definido en la especificación TTCN-3.

– `equals(TriComponentId component)`
Compara si son iguales `component` y este `TriComponentId`. Devuelve `true` si los dos componentes tienen la misma representación de este identificador de componente único y `false` en caso contrario.

#### 6.3.2.4 TriComponentIdListType

**`TriComponentIdListType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriComponentIdListType
package org.etsi.ttcn.tri;
public interface TriComponentIdListType {
    public int size();
    public boolean isEmpty();
    public java.util.Enumeration getComponents();
    public TriComponentId get(int index);
    public void clear();
    public void add(TriComponentId comp);
}
```

##### 6.3.2.4.1 Métodos

`size()`
Devuelve el número de componentes que contiene esta lista.

`isEmpty()`
Devuelve `true` si la lista no contiene componentes.

`getComponents()`
Devuelve una `Enumeration` de los componentes que contiene la lista. Los componentes de la enumeration figuran en el mismo orden en el que aparecen en la lista.

`get(int index)`
Devuelve el `TriComponentId` en la posición especificada.

`clear()`
Suprime todos los componentes de este `TriComponentIdList`.

`add(TriComponentId comp)`
Añade `comp` al final de esta `TriComponentIdList`.

#### 6.3.2.5 TriMessageType

**`TriMessageType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriMessageType
package org.etsi.ttcn.tri;
public interface TriMessage {
    public byte[] getEncodedMessage();
    public void setEncodedMessage(byte[] message);
    public boolean equals(TriMessage message);
}
```

##### 6.3.2.5.1 Métodos

- `getEncodedMessage()`
Devuelve el mensaje codificado conforme a las reglas de codificación definidas en la especificación TTCN3.

- `setEncodedMessage(byte[] message)`
Asigna el valor `message` a la representación de mensaje codificada de este `TriMessage`.

- `equals(TriMessage message)`
Compara si son iguales `message` y `TriMessage`. Devuelve `true` si tienen la misma representación codificada y `false` en caso contrario.

#### 6.3.2.6 TriAddressType

**`TriAddressType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriAddressType
package org.etsi.ttcn.tri;
public interface TriAddress {
    public byte[] getEncodedAddress();
    public void setEncodedAddress(byte[] address);
    public boolean equals(TriAddress address);
}
```

#### 6.3.2.6.1 Métodos

- `getEncodedAddress()`
  Devuelve la dirección codificada.

- `setEncodedAddress(byte[] address)`
  Asigna el valor `address` a la dirección codificada de este `TriAddress`.

- `equals(TriAddress address)`
  Comprueba si son iguales `address` y este `TriAddress`. Devuelve `true` si tienen la misma representación codificada, y `false` en caso contrario.

### 6.3.2.7 TriAddressListType

**`TriAddressListType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriAddressListType
package org.etsi.ttcn.tri;
public interface TriAddressListType {
    public int size();
    public boolean isEmpty();
    public java.util.Enumeration getAddresses();
    public TriAddress get(int index);
    public void clear();
    public void add(TriAddress addr);
}
```

#### 6.3.2.7.1 Métodos

`size()`
  Devuelve el número de componentes que contiene esta lista.

`isEmpty()`
  Devuelve `true` si la lista no contiene componentes.

`getAddresses()`
  Devuelve una `Enumeration` de los componentes que contiene esta lista. Las direcciones de la enumeration figuran en el mismo orden en el que aparecen en la lista.

`get(int index)`
  Devuelve la `TriAddress` en la posición especificada.

`clear()`
  Suprime todas las direcciones de esta `TriAddressList`.

`add(TriAddress addr)`
  Añade `addr` al `final` de esta `TriAddressList`.

### 6.3.2.8 TriSignatureIdType

**`TriSignatureIdType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriSignatureIdType
package org.etsi.ttcn.tri;
public interface TriSignatureId {
    public String getSignatureName();
    public void setSignatureName(String sigName);
    public boolean equals(TriSignatureId sig);
}
```

#### 6.3.2.8.1 Métodos

- `getSignatureName()`
  Devuelve el identificador de signatura definido en la especificación TTCN-3.

- `setSignatureName(String sigName)`
  Asigna el valor `sigName` al identificador de este `TriSignatureId`.

- `equals(TriSignatureId sig)`
  Compara si son iguales `sig` y este `TriSignatureId`. Devuelve `true` si las dos signaturas tienen el mismo identificador de signatura y `false` en caso contrario.

#### 6.3.2.9 TriParameterType

**`TriParameterType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriParameterType
package org.etsi.ttcn.tri;
public interface TriParameter {
    public String getParameterName();
    public void setParameterName(String name);
    public int getParameterPassingMode();
    public void setParameterPassingMode(in mode);
    public byte[] getEncodedParameter();
    public void setEncodedParameter(byte[] parameter);
}
```

##### 6.3.2.9.1 Métodos

- `getParameterName()`
  Devuelve el nombre del parámetro definido en la especificación TTCN-3.

- `setParameterName(String name)`
  Asigna el valor de name al nombre de este `TriParameter`.

- `getParameterPassingMode()`
  Devuelve el modo de pasar parámetros de este parámetro.

- `setParameterPassingMode(in mode)`
  Asigna el valor `mode` al modo de pasar parámetros de este `TriParameter`

- `getEncodedParameter()`
  Devuelve la representación del parámetro codificada de este `TriParameter`, o el objeto `null` si el parámetro contiene el valor distintivo `null` (véase también 5.5.4.1).

- `setEncodedParameter(byte[] parameter)`
  Asigna el valor `parameter` a la representación del parámetro codificado de este `TriParameter`. Si para indicar que este parámetro no contiene valor alguno se asigna el valor distintivo `null` deberá pasarse como `parameter` el valor `null` en Java (véase también 5.5.4.1).

#### 6.3.2.10 TriParameterPassingModeType

**`TriParameterPassingModeType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriParameterPassingModeType
package org.etsi.ttcn.tri;
public interface TriParameterPassingMode {
    public final static int TRI_IN = 0;
    public final static int TRI_INOUT = 1;
    public final static int TRI_OUT = 2;
}
```

##### 6.3.2.10.1 Constantes

- `TRI_IN`
  Se utiliza para indicar que este `TriParameter` es un parámetro `in`.

- `TRI_INOUT`
  Se utiliza para indicar que este `TriParameter` es un parámetro `inout`.

- `TRI_OUT`
  Se utiliza para indicar que este `TriParameter` es un parámetro `out`.

#### 6.3.2.11 TriParameterListType

**`TriParameterListType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriParameterListType
package org.etsi.ttcn.tri;
public interface TriParameterList {
    public int size();
    public boolean isEmpty();
    public java.util.Enumeration getParameters();
    public TriParameter get(int index);
    public void clear();
    public void add(TriParameter parameter);
}
```

#### 6.3.2.11.1 Métodos

`size()`
Devuelve el número de parámetros que contiene esta lista.

`isEmpty()`
Devuelve `true` si la lista no contiene parámetros.

`getParameters()`
Devuelve una `Enumeration` de los parámetros que contiene la lista. Los parámetros de enumeration figuran en el mismo orden en el que aparecen en la lista.

`get(int index)`
Devuelve el `TriParameter` en la posición especificada.

`clear()`
Suprime todos los parámetros de esta `TriParameterList`.

`add(TriParameter parameter)`
Añade `parameter` al final de esta `TriParameterList`.

### 6.3.2.12 TriExceptionType

**`TriExceptionType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriExceptionType
package org.etsi.ttcn.tri;
public interface TriException {
    public byte[] getEncodedException();
    public void setEncodedException(byte[] message);
    public boolean equals(TriException exc);
}
```

#### 6.3.2.12.1 Métodos

`- getEncodedException()`
Devuelve la excepción codificada conforme a las reglas de codificación definidas en la especificación TTCN-3.

`- setEncodedMessage(byte[] exc)`
Asigna el valor `exc` a la representación de la sección codificada de esta `TriException`.

`- equals(TriException exc)`
Compara si son iguales `exc` y esta `TriException`. Devuelve `true` si las dos excepciones tienen la misma representación codificada y `false` en caso contrario.

### 6.3.2.13 TriTimerIdType

**`TriTimerIdType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriTimerIdType
package org.etsi.ttcn.tri;
public interface TriTimerId {
    public String getTimerName();
    public boolean equals(TriTimerId timer);
}
```

#### 6.3.2.13.1 Métodos

`- getTimerName()`
Devuelve el nombre de este identificador temporizador definido en la especificación TTCN-3. Cuando se trata de un temporizador implícito, el resultado depende de la implementación (véase 4.1.2).

`- equals(TriTimerId timer)`
Compara si son iguales `timer` y este `TriTimerId`. Devuelve `true` si los dos identificadores representan al mismo temporizador y `false` en caso contrario.

### 6.3.2.14 TriTimerDurationType

**`TriTimerDurationType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriTimerDurationType
package org.etsi.ttcn.tri;
public interface TriTimerDuration {
    public double getDuration();
    public void setDuration(double duration);
    public boolean equals(TriTimerDuration duration);
}
```

#### 6.3.2.14.1 Métodos

- `getDuration()`
  Devuelve la duración del temporizador como un tipo `double`.

- `setDuration(double duration)`
  Asigna a la duración de esta `TriTimerDuration` el valor `duration`.

- `equals(TriTimerDuration duration)`
  Compara si son iguales duration y esta `TriTimerDuration`. Devuelve `true` si y sólo si los dos tienen la misma duración y `false` en caso contrario.

### 6.3.2.15 TriFunctionIdType

**`TriFunctionIdType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriFunctionIdType
package org.etsi.ttcn.tri;
public interface TriFunctionId {
    public String toString();
    public String getFunctionName();
    public boolean equals(TriFunctionId fun);
}
```

#### 6.3.2.15.1 Métodos

- `toString()`
  Devuelve la representación en cadena de la función definida en la especificación TTCN-3.

- `getFunctionName()`
  Devuelve el identificador de la función definida en la especificación TTCN-3.

- `equals(TriFunctionId fun)`
  Compara si son iguales `fun` y este `TriFunctionId`. Devuelve `true` si las dos funciones tienen el mismo identificador de función y `false` en caso contrario.

### 6.3.2.16 TriTestCaseIdType

**`TriTestCaseIdType`** corresponde con la siguiente interfaz:

```
// TRI IDL TriTestCaseIdType
package org.etsi.ttcn.tri;
public interface TriTestCaseId {
    public String toString();
    public String getTestCaseName();
    public boolean equals(TriTestCaseId tc);
}
```

#### 6.3.2.16.1 Métodos

- `toString()`
  Devuelve la representación en cadena del caso de prueba definida en la especificación TTCN-3.

- `getTestCaseName()`
  Devuelve el identificador de caso de prueba definido en la especificación TTCN-3.

- `equals(TriTestCaseId tc)`
  Compara si son iguales `tc` y este `TriTestCaseId`. Devuelve `true` si los dos casos de prueba tienen el mismo identificador y `false` en caso contrario.

### 6.3.2.17 TriActionTemplateType

Obsoleto.

#### 6.3.2.18 TriStatusType

**`TriStatusType`** corresponde con la siguiente interfaz:

```
// TriStatusType
package org.etsi.ttcn.tri;
public interface TriStatus {
    public final static int TRI_OK = 0;
    public final static int TRI_ERROR = -1;
    public String toString();
    public int getStatus();
    public void setStatus(int status);
    public boolean equals(TriStatus status);
}
```

##### 6.3.2.18.1 Métodos

- `toString()`
  Devuelve el estado expresado como una cadena.

- `getStatus()`
  Devuelve el estado de este `TriStatus`.

- `setStatus(int status)`
  Asigna el estado de este `TriStatus`.

- `equals(TriStatus status)`
  Compara si son iguales `status` y `TriStatus`. Devuelve `true` si los dos tienen el mismo estado y `false` en caso contrario.

### 6.4 Constantes

En esta correspondencia con el lenguaje Java se han especificado constantes. Todas las constantes se definen como `public final static` y son accesibles desde cualquier objeto de todos los lotes. Las constantes definidas en esta cláusula no están definidas en la cláusula IDL, sino que son el resultado de la especificación de los tipos IDL de la TRI marcados como nativos.

Las siguientes constantes pueden utilizase para determinar el modo de pasar parámetros de los parámetros TTCN-3 (véase también 6.3.2.10).

- org.etsi.ttcn.tri.TriParameterPassingMode.TRI_IN;
- org.etsi.ttcn.tri.TriParameterPassingMode.TRI_INOUT;
- org.etsi.ttcn.tri.TriParameterPassingMode.TRI_OUT.

Los valores de ejemplares de estas constantes indicarán el modo de pasar parámetros definido en las signaturas de procedimiento TTCN-3.

En el caso del valor de parámetro distintivo `null`, el valor de parámetro codificado deberá ser `null` en Java.

Las siguientes constantes deberán utilizarse para indicar que el método se ha ejecutado correctamente (véase también 6.3.2.18):

- org.etsi.ttcn.tri.TriStatus.TRI_OK;
- org.etsi.ttcn.tri.TriStatus.TRI_ERROR.

### 6.5 Correspondencia de interfaces

En el IDL de la TRI se definen dos interfaces, a saber, **`triCommunication`** y **`triPlatform`**. Dado que las operaciones se definen para diferentes direcciones dentro de esta interfaz, es decir, algunas operaciones sólo pueden llamarse desde la TTCN-3 ejecutable (TE) al adaptador de sistema (SA), mientras que otras sólo pueden llamarse desde el SA a la TE. Para explicar este hecho se dividen las interfaces TRI IDL en dos subinterfaces, en las que la entidad llamada aparece como sufijo.

**Cuadro 4/Z.144 – Subinterfaces**

| Llamante/llamado | TE | SA | PA |
|---|---|---|---|
| TE | - | TriCommunicationSA | triPlatformPA |
| SA | TriCommunicationTE | - | - |
| PA | TriPlatformTE | - | - |

Todos los métodos definidos en estas interfaces deben comportarse como se describe en la cláusula 5.

### 6.5.1 Modo de pasar parámetros Out e InOut

Para el modo de pasar parámetros `out` o `inout` se utilizan los siguientes tipos IDL:

- TriParameter.
- TriParameterList.
- TriBoolean.
- TriTimerDuration.

En caso de que se utilicen en el modo de pasar parámetros `out` o `inout`, los objetos de la clase correspondiente se pasarán con la llamada al método. La entidad llamada puede acceder luego a los métodos para modificar los valores que devuelve.

### 6.5.2 Interfaz triCommunication

La interfaz **`triCommunication`** se divide en dos subinterfaces, a saber `triCommunicationSA`, para llamadas desde la TE al SA, y `triCommunicationTE`, para llamadas desde SA a la TE.

#### 6.5.2.1 triCommunicationSA

La interfaz `triCommunicationSA` corresponde con la siguiente interfaz:

```
// TriCommunication
// TE -> SA
package org.etsi.ttcn.tri;
public interface TriCommunicationSA {
    // Reset Operation
    // Ref: TRI-Definition 5.5.1
    TriStatus triSAReset();

    // Connection handling operations
    // Ref: TRI-Definition 5.5.2.1
    public TriStatus triExecuteTestCase(TriTestCaseId
        testCaseId,TriPortIdList tsiPorts);
    // Ref: TRI-Definition 5.5.2.2
    public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId);
    // Ref: TRI-Definition 5.5.2.3
    public TriStatus triUnmap(TriPortId compPortId, TriPortId tsiPortId);

    // Message based communication operations
    // Ref: TRI-Definition 5.5.3.1
    public TriStatus triSend(TriComponentId componentId, TriPortId tsiPortId,
            TriAddress sutAddress, TriMessage sendMessage);
    // Ref: TRI-Definition 5.5.3.2
    public TriStatus triSendBC(TriComponentId componentId, TriPortId tsiPortId,
            TriMessage sendMessage);
    // Ref: TRI-Definition 5.5.3.3
    public TriStatus triSendMC(TriComponentId componentId, TriPortId tsiPortId,
            TriAddressList addresses, TriMessage sendMessage);

    // Procedure based communication operations
    // Ref: TRI-Definition 5.5.4.1
    public TriStatus triCall(TriComponentId componentId,
            TriPortId tsiPortId, TriAddress sutAddress,
            TriSignatureId signatureId, TriParameterList parameterList);
    // Ref: TRI-Definition 5.5.4.2
    public TriStatus triCallBC(TriComponentId componentId,
            TriPortId tsiPortId,
            TriSignatureId signatureId, TriParameterList parameterList);
    // Ref: TRI-Definition 5.5.4.3
    public TriStatus triCallMC(TriComponentId componentId,
            TriPortId tsiPortId, TriAddressList sutAddresses,
            TriSignatureId signatureId, TriParameterList parameterList);
```

```
    // Ref: TRI-Definition 5.5.4.4
    public TriStatus triReply(TriComponentId componentId,
            TriPortId tsiPortId, TriAddress sutAddress,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);
    // Ref: TRI-Definition 5.5.4.5
    public TriStatus triReplyBC(TriComponentId componentId,
            TriPortId tsiPortId,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);
    // Ref: TRI-Definition 5.5.4.6
    public TriStatus triReplyMC(TriComponentId componentId,
            TriPortId tsiPortId, TriAddressList sutAddresses,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);

    // Ref: TRI-Definition 5.5.4.7
    public TriStatus triRaise(TriComponentId componentId, TriPortId tsiPortId,
            TriAddress sutAddress,
            TriSignatureId signatureId,
            TriException exc);
    // Ref: TRI-Definition 5.5.4.8
    public TriStatus triRaiseBC(TriComponentId componentId, TriPortId tsiPortId,
            TriSignatureId signatureId,
            TriException exc);
    // Ref: TRI-Definition 5.5.4.9
    public TriStatus triRaiseMC(TriComponentId componentId,TriPortId tsiPortId,
            TriAddresses sutAddresses,
            TriSignatureId signatureId,
            TriException exc);

    // Miscellaneous operations
    // Ref: TRI-Definition 5.5.5.1
    public TriStatus triSutActionInformal(String description);

}
```

#### 6.5.2.2 triCommunicationTE

La interfaz `triCommunicationTE` corresponde con la siguiente interfaz:

```
// TriCommunication
// SA -> TE
package org.etsi.ttcn.tri;
public interface TriCommunicationTE {
    // Message based communication operations
    // Ref: TRI-Definition 5.5.3.4
    public void triEnqueueMsg(TriPortId tsiPortId,
            TriAddress sutAddress, TriComponentId componentId,
            TriMessage receivedMessage);

    // Procedure based communication operations
    // Ref: TRI-Definition 5.5.4.10
    public void triEnqueueCall(TriPortId tsiPortId,
            TriAddress SUTaddress, TriComponentId componentId,
            TriSignatureId signatureId, TriParameterList parameterList );

    // Ref: TRI-Definition 5.5.4.11
    public void triEnqueueReply(TriPortId tsiPortId, TriAddress sutAddress,
            TriComponentId componentId, TriSignatureId signatureId,
            TriParameterList parameterList, TriParameter returnValue);

    // Ref: TRI-Definition 5.5.4.12
    public void triEnqueueException(TriPortId tsiPortId,
            TriAddress sutAddress, TriComponentId componentId,
        TriSignatureId signatureId, TriException exc);
}
```

### 6.5.3 Interfaz triPlatform

La interfaz **`triPlatform`** se divide en dos subinterfaces, a saber, `triPlatformPA`, para llamadas de la TE al PA y `triPlatformTE`, para llamadas del PA a la TE.

#### 6.5.3.1 TriPlatformPA

La interfaz `triPlatformPA` corresponde con la siguiente interfaz:

```
// TriPlatform
// TE -> PA
package org.etsi.ttcn.tri;
public interface TriPlatformPA {
    // Ref: TRI-Definition 5.6.1
    public TriStatus triPAReset();

    // Timer handling operations
    // Ref: TRI-Definition 5.6.2.1
    public TriStatus triStartTimer(TriTimerId timerId,
        TriTimerDuration timerDuration);

    // Ref: TRI-Definition 5.6.2.2
    public TriStatus triStopTimer(TriTimerId timerId);

    // Ref: TRI-Definition 5.6.2.3
    public TriStatus triReadTimer(TriTimerId timerId,
        TriTimerDuration elapsedTime);

    // Ref: TRI-Definition 5.6.2.4
    public TriStatus triTimerRunning(TriTimerId timerId,
        TriBoolean running);

    // Miscellaneous operations

    // Ref: TRI-Definition 5.6.3.1
    public TriStatus triExternalFunction(TriFunctionId functionId,
     TriParameterList parameterList, TriParameter returnValue);
}
```

#### 6.5.3.2 TriPlatformTE

La interfaz `triPlatformTE` corresponde con la siguiente interfaz Java:

```
// TriPlatform
// PA -> TE
package org.etsi.ttcn.tri;
public interface TriPlatformTE {
    // Ref: TRI-Definition 5.6.2.5
    public void triTimeout(TriTimerId timerId);
}
```

## 6.6 Parámetros opcionales

Según se dijo en la cláusula 5.4, se utilizará un valor reservado para indicar la ausencia de un parámetro opcional. En el caso del lenguaje Java, deberá utilizarse el valor `null` en Java para indicar la ausencia de un valor opcional. Por ejemplo, si en la operación `triSend` se omite el parámetro address, se deberá llamar a la operación del modo siguiente: `triSend(componentId, tsiPortId, null, sendMessage)`.

## 6.7 Inicialización de la TRI

Todos los métodos son no estáticos, es decir operaciones que sólo pueden aplicarse a objetos. Dado que en esta Recomendación no se definen estrategias de implementación concretas de la TE, el SA y el PA, queda fuera del alcance de la presente Recomendación el mecanismo por el cual la TE, la SA, o el PA averiguan cómo utilizar los correspondientes objetos.

Los fabricantes de herramientas deberán proporcionar métodos a los programadores de SA y el PA para registrar la TE, el SA y el PA en sus respectivos interlocutores de la comunicación.

## 6.8 Gestión de errores

En esta relación con el lenguaje Java no se define ninguna gestión de errores diferente a la definida en 5.2. En particular, no se definen mecanismos de gestión de excepciones.

# 7 Correspondencia con el lenguaje ANSI-C

## 7.1 Introducción

En la presente cláusula se define la correspondencia de la TRI con el lenguaje ANSI-C y los tipos de datos abstractos especificados en 5.3. Para los tipos IDL básicos, esta correspondencia es conforme con las recomendaciones OMG.

## 7.2 Nombres y alcances

Los identificadores de parámetros en C deben comenzar en minúsculas y la otra parte integrante del identificador de parámetro debe comenzar en mayúsculas. Por ejemplo, el parámetro IDL `SUTaddress` se escribe con `sutAddress` en C.

En los identificadores de tipos de datos abstractos en C se omite el sufijo `Type` utilizado en la definición IDL. Por ejemplo, el tipo IDL `TriPortIdType` corresponde a `TriPortId` en C.

Las especificaciones de C antiguas limitan a 8 el número de caracteres más significativos de un identificador único. No obstante, las especificaciones ANSI-C más recientes permiten hasta 31 caracteres más significativos. Aparte de este problema, no existen otros problemas de denominación o alcance en esta correspondencia.

### 7.2.1 Correspondencia de tipos abstractos

| TRI ADT | Representación ANSI-C | Notas y comentarios |
|---|---|---|
| `TriAddress` | `BinaryString` | |
| `TriAddressList` | `typedef struct TriAddressList`<br>`{`<br>` TriAddress** addrList;`<br>` long int length;`<br>`} TriAddressList;` | NOTA – No existen valores especiales que delimiten el final de `addrList[]`. El campo `length` se utiliza para desplazarse por este arreglo adecuadamente. |
| `TriComponentId` | `typedef struct TriComponentId`<br>`{`<br>` BinaryString compInst;`<br>` String compName;`<br>` QualifiedName compType;`<br>`} TriComponentId;` | NOTA – `compInst` es el ejemplar del componente. |
| `TriComponentIdList` | `typedef struct TriComponentIdList`<br>`{`<br>` TriComponentId** compIdList;`<br>` long int length;`<br>`} TriComponentIdList;` | NOTA – No existen valores especiales que delimiten el final de `compIdList[]`. El campo `length` se utiliza para desplazarse por este arreglo adecuadamente. |
| `TriException` | `BinaryString` | |
| `TriFunctionId` | `QualifiedName` | |
| `TriMessage` | `BinaryString` | |
| `TriParameterList` | `typedef struct TriParameterList`<br>`{`<br>` TriParameter** parList;`<br>` long int length;`<br>`} TriParameterList;` | NOTA – No existen valores especiales que delimiten el final de `parList`. El campo `length` se utiliza para desplazarse por este arreglo adecuadamente. |
| `TriParameter` | `typedef struct TriParameter`<br>`{`<br>` BinaryString par;`<br>` TriParameterPassingMode mode;`<br>`} TriParameter;` | |

| TRI ADT | Representación ANSI-C | Notas y comentarios |
|---|---|---|
| TriParameterPassingMode | typedef enum<br>{<br>TRI_IN = 0,<br>TRI_INOUT = 1,<br>TRI_OUT = 2<br>} TriParameterPassingMode; | NOTA – Los valores de los ejemplares de este tipo indicarán el modo de pasar parámetros definido en las correspondientes signaturas de procedimiento TTCN-3. |
| TriPortIdList | typedef struct TriPortIdList<br>{<br>TriPortId** portIdList;<br>long int length;<br>} TriPortIdList; | NOTA – No existen valores especiales que delimiten el final de `portIdList[]`. El campo `length` se utiliza para desplazarse por este arreglo adecuadamente. |
| TriPortId | typedef struct TriPortId<br>{<br>TriComponentId compInst;<br>char* portName;<br>long int portIndex;<br>QualifiedName portType;<br>void* aux;<br>} TriPortId; | NOTA 1 – `compInst` es el ejemplar del componente.<br>NOTA 2 – Para una declaración particular (distinta de un arreglo) el valor `portIndex` debe ser –1.<br>NOTA 3 – El campo `aux` está reservado para la futura ampliación de la funcionalidad TRI. |
| TriSignatureId | QualifiedName | |
| TriStatus | long int<br>#define TRI_ERROR -1<br>#define TRI_OK 0 | NOTA – Todo valor negativo se reserva para la futura ampliación de la funcionalidad TRI. |
| TriTestCaseId | QualifiedName | |
| TriTimerDuration | Double | |
| TriTimerId | BinaryString | NOTA – Las declaraciones pendientes en el temporizador y la semántica del estado instantáneo puede afectar a la futura representación. |

### 7.2.2 Definiciones de tipo ANSI-C

| C ADT | Definición de tipo | Notas y comentarios |
|---|---|---|
| BinaryString | typedef struct BinaryString<br>{<br>unsigned char* data;<br>long int bits;<br>void* aux;<br>} BinaryString; | NOTA 1 – `data` es una cadena que no termina en null.<br>NOTA 2 – `bits` es el número de bits utilizado en los datos. El valor –1 se utiliza para indicar un valor omitido.<br>NOTA 3 – El campo `aux` está reservado para la futura ampliación de la funcionalidad TRI. |
| QualifiedName | typedef struct QualifiedName<br>{<br>char* moduleName;<br>char* objectName;<br>void* aux;<br>} QualifiedName; | NOTA 1 – Los campos `moduleName` y `objectName` son los identificadores TTCN-3 literalmente.<br>NOTA 2 – El campo `aux` está reservado para la futura ampliación de la funcionalidad TRI. |

### 7.2.3 Correspondencia del tipo IDL

| Tipo IDL | Representación ANSI-C | Notas y comentarios |
|---|---|---|
| Boolean | unsigned char | Correspondencia entre OMG IDL y C++ |
| String | char* | Correspondencia entre OMG IDL y C++ |

### 7.2.4 Correspondencia con las operaciones TRI

| Representación IDL | Representación ANSI-C |
|---|---|
| TriStatusType triSAReset() | TriStatus triSAReset() |
| TriStatusType triExecuteTestCase<br>(in TriTestCaseIdType testCaseId,<br>in TriPortIdListType tsiPortList) | TriStatus triExecuteTestCase<br>(const TriTestCaseId* testCaseId,<br>const TriPortIdList* tsiPortList) |
| TriStatusType triMap<br>(in TriPortIdType compPortId,<br>in TriPortIdType tsiPortId) | TriStatus triMap<br>(const TriPortId* compPortId,<br>const TriPortId* tsiPortId) |
| TriStatusType triUnmap<br>(in TriPortIdType compPortId,<br>in TriPortIdType tsiPortId) | TriStatus triUnmap<br>(const TriPortId* compPortId,<br>const TriPortId* tsiPortId) |
| TriStatusType triSend<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriMessageType sendMessage) | TriStatus triSend<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriMessage* sendMessage) |
| TriStatusType triSendBC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriMessageType sendMessage) | TriStatus triSendBC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriMessage* sendMessage) |
| TriStatusType triSendMC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressListType SUTaddresses,<br>in TriMessageType sendMessage) | TriStatus triSendMC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddressList* sutAddresses,<br>const TriMessage* sendMessage) |
| void triEnqueueMsg<br>(in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriComponentIdType componentId,<br>in TriMessageType receivedMessage) | void triEnqueueMsg<br>(const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriComponentId* componentId,<br>const TriMessage* receivedMessage) |
| TriStatusType triCall<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList) | TriStatus triCall<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList) |
| TriStatusType triCallBC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList) | TriStatus triCallBC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList) |
| TriStatusType triCallMC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressListType SUTaddresses,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList) | TriStatus triCallMC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddressList* sutAddresses,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList) |

| Representación IDL | Representación ANSI-C |
|---|---|
| TriStatusType triReply<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList,<br>in TriParameterType returnValue) | TriStatus triReply<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList,<br>const TriParameter* returnValue) |
| TriStatusType triReplyBC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList,<br>in TriParameterType returnValue) | TriStatus triReplyBC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList,<br>const TriParameter* returnValue) |
| TriStatusType triReplyMC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressListType SUTaddresses,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList,<br>in TriParameterType returnValue) | TriStatus triReplyMC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddressList* sutAddresses,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList,<br>const TriParameter* returnValue) |
| TriStatusType triRaise<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriSignatureIdType signatureId,<br>in TriExceptionType exc) | TriStatus triRaise<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriSignatureId* signatureId,<br>const TriException* exception) |
| TriStatusType triRaiseBC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriSignatureIdType signatureId,<br>in TriExceptionType exc) | TriStatus triRaiseBC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriSignatureId* signatureId,<br>const TriException* exception) |
| TriStatusType triRaiseMC<br>(in TriComponentIdType componentId,<br>in TriPortIdType tsiPortId,<br>in TriAddressListType SUTaddresses,<br>in TriSignatureIdType signatureId,<br>in TriExceptionType exc) | TriStatus triRaiseMC<br>(const TriComponentId* componentId,<br>const TriPortId* tsiPortId,<br>const TriAddressList* sutAddresses,<br>const TriSignatureId* signatureId,<br>const TriException* exception) |
| void triEnqueueCall<br>(in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriComponentId componentId,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList) | void triEnqueueCall<br>(const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriComponentId* componentId,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList) |
| void triEnqueueReply<br>(in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriComponentIdType componentId,<br>in TriSignatureIdType signatureId,<br>in TriParameterListType parameterList,<br>in TriParameterType returnValue) | void triEnqueueReply<br>(const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriComponentId* componentId,<br>const TriSignatureId* signatureId,<br>const TriParameterList* parameterList,<br>const TriParameter* returnValue) |

| Representación IDL | Representación ANSI-C |
|---|---|
| void triEnqueueException<br>(in TriPortIdType tsiPortId,<br>in TriAddressType SUTaddress,<br>in TriComponentIdType componentId,<br>in TriSignatureIdType signatureId,<br>in TriExceptionType exc) | void triEnqueueException<br>(const TriPortId* tsiPortId,<br>const TriAddress* sutAddress,<br>const TriComponentId* componentId,<br>const TriSignatureId* signatureId,<br>const TriException* exception) |
| TriStatusType triSUTActionInformal<br>(in string description) | TriStatus triSUTActionInformal<br>(const char* description) |
| TriStatusType triPAReset() | TriStatus triPAReset() |
| TriStatusType triStartTimer<br>(in TriTimerIdType timerId,<br>in TriTimerDurationType timerDuration) | TriStatus triStartTimer<br>(const TriTimerId* timerId,<br>TriTimerDuration timerDuration) |
| TriStatusType triStopTimer<br>(in TriTimerIdType timerId) | TriStatus triStopTimer<br>(const TriTimerId* timerId) |
| TriStatusType triReadTimer<br>(in TriTimerIdType timerId,<br>out TriTimerDurationType elapsedTime) | TriStatus triReadTimer<br>(const TriTimerId* timerId,<br>TriTimerDuration* elapsedTime) |
| TriStatusType triTimerRunning<br>(in TriTimerIdType timerId,<br>out boolean running) | TriStatus triTimerRunning<br>(const TriTimerId* timerId,<br>unsigned char* running) |
| void triTimeout<br>(in TriTimerIdType timerId) | void triTimeout<br>(const TriTimerId* timerId) |
| TriStatusType triExternalFunction<br>(in TriFunctionIdType functionId,<br>inout TriParameterListType parameterList,<br>out TriParameterType returnValue) | TriStatus triExternalFunction<br>(const TriFunctionId* functionId,<br>TriParameterList* parameterList,<br>TriParameter* returnValue) |

### 7.3 Gestión de memoria

Obsoleto.

### 7.4 Gestión de errores

No se ha definido la gestión de errores para esta correspondencia.

## 8 Escenarios de utilización

En la presente cláusula se describen escenarios de utilización que podrían ayudar a los usuarios de la TRI y a los fabricantes de herramientas a comprender la semántica TRI de las operaciones definidas en esta Recomendación.

Los escenarios se describen mediante diagramas secuenciales de mensajes (MSC). Cada ejemplo consiste en un fragmento de código TTCN-3 que utiliza funciones de comunicación TTCN-3 con el SUT y funciones de gestión de temporizadores. Los diagramas MSC describen las interacciones entre las entidades TE, SA y PA con el SUT.

Obsérvese que los fragmentos de código TTCN-3 no están completos, dado que el principal objetivo de éstos es mostrar la utilización de un comportamiento dinámico. Todos los ejemplos que figuran a continuación utilizan una secuencia preámbulo común de operaciones TRI, la cual se muestra en la figura 2.

Cabe observar que los diagramas MSC que contiene esta cláusula utilizan pares de mensajes para modelizar cada operación TRI. El mensaje MSC triMap seguido de triMapOK indica, por ejemplo, que la TE ha llamado a la operación triMap de la TRI y ésta se ha llevado a cabo correctamente desde el SA. Las llamadas de operaciones TRI se muestran mediante valores y tipos abstractos y se incluyen únicamente a título ilustrativo. La representación concreta de esos parámetros en un determinado lenguaje se define en la correspondiente correspondencia lingüística.

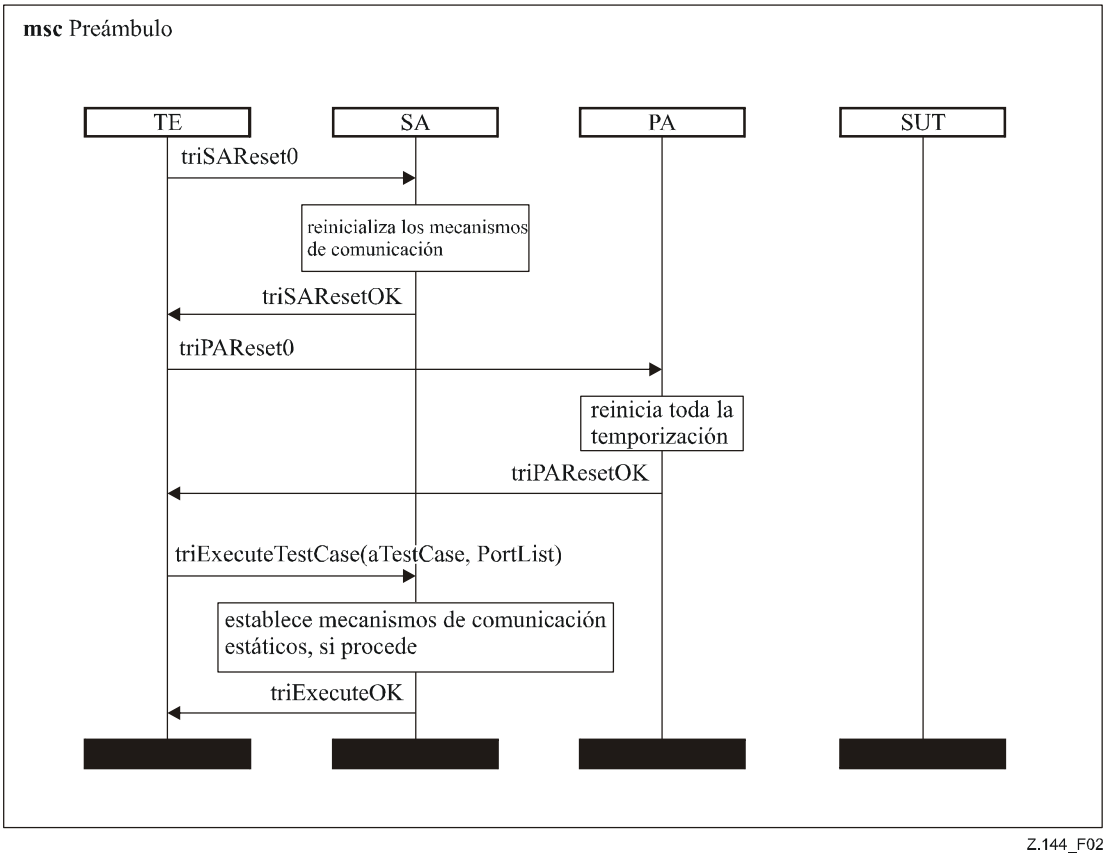


**Figura 2/Z.144 – Preámbulo MSC común**

## 8.1 Primer escenario

En este primer escenario se muestran algunas de las operaciones de temporización TTCN-3, por ejemplo la puesta en marcha y determinación del estado del temporizador, operaciones de comunicación basadas en mensajes, por ejemplo enviar y recibir, y operaciones de gestión de conexión, por ejemplo, hacer y deshacer la correspondencia.

### 8.1.1 Fragmento TTCN-3

```
module triScenario1
{
 external function MyFunction();

 type port PortTypeMsg message { inout integer }

 type component MyComponent {
    port PortTypeMsg MyPort;
    timer MyTimer
 }

 type component MyTSI {
    port PortTypeMsg PCO1;
 }

 testcase scenario1() runs on MyComponent system MyTSI
 {
    MyPort.clear;
    MyPort.start;
    MyTimer.start(2);

    map(MyComponent: MyPort, system: PCO1);
    MyPort.send (integer : 5);
    if (MyTimer.running)
    {
     MyPort.receive(integer:7);
    }
    else
    {
     MyFunction();
    }
```

```
    unmap(MyComponent: MyPort, system:PCO1);
    MyPort.stop;
 }

 control {
    execute( scenario1() );
 }

}
```

### 8.1.2 Diagrama secuencial de mensajes

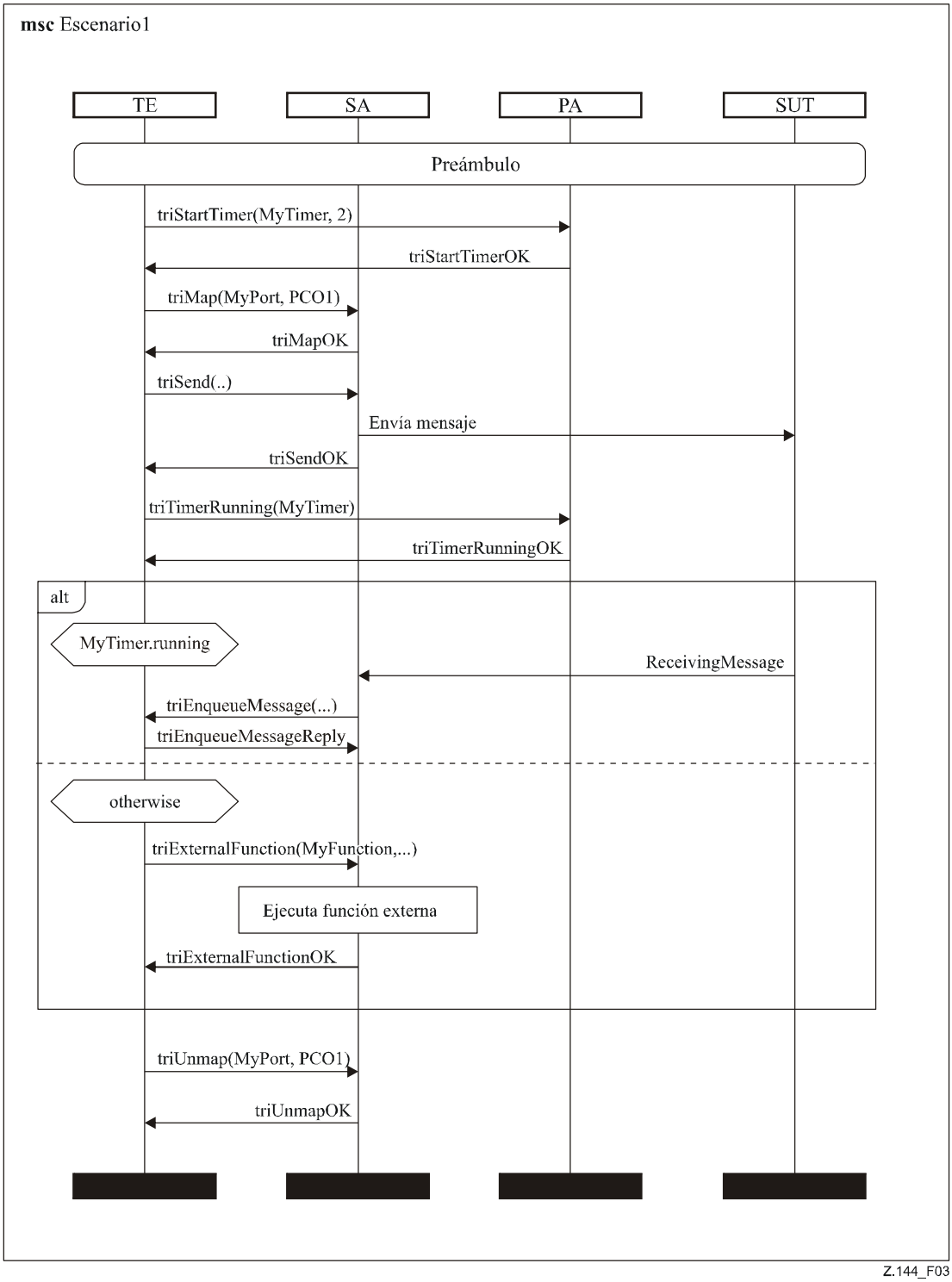


**Figura 3/Z.144 – Escenario de utilización 1**

## 8.2 Segundo escenario

En este segundo escenario se muestra un caso similar al anterior en el que se utilizan además operaciones de comunicación basadas en procedimientos sincronizados que inicia el componente de prueba `MyComponent`. En este ejemplo se supone que `MyComponent` se comporta como un MTC.

### 8.2.1 Fragmento TTCN-3

```
module triScenario2
{

 signature MyProc ( in float par1, inout float par2)
    exception(MyExceptionType);

 type record MyExceptionType { FieldType1 par1, FieldType2 par2 }

 type port PortTypeProc procedure { out MyProc }

 type component MyComponent {
    port PortTypeProc MyPort;
    timer MyTimer = 7
 }

 testcase scenario2() runs on MyComponent
 {
    var float MyVar;

    MyPort.clear;
    MyPort.start;
    MyTimer.start;

 MyVar := MyTimer.read;

 if (MyVar>5.0) {
     MyPort.call (MyProc:{MyVar, 5.7}, 5);
     alt {
     [] MyPort.getreply(MyProc:{-,MyVar*5}) {}
     [] MyPort.catch (MyProc, MyExceptionType:* ) {}
     [] MyPort.catch (timeout) {}
     }
    }
MyTimer.stop;
    MyPort.stop;
 }

 control {
    execute( scenario2() );
 }

}
```

### 8.2.2 Diagrama secuencial de mensajes

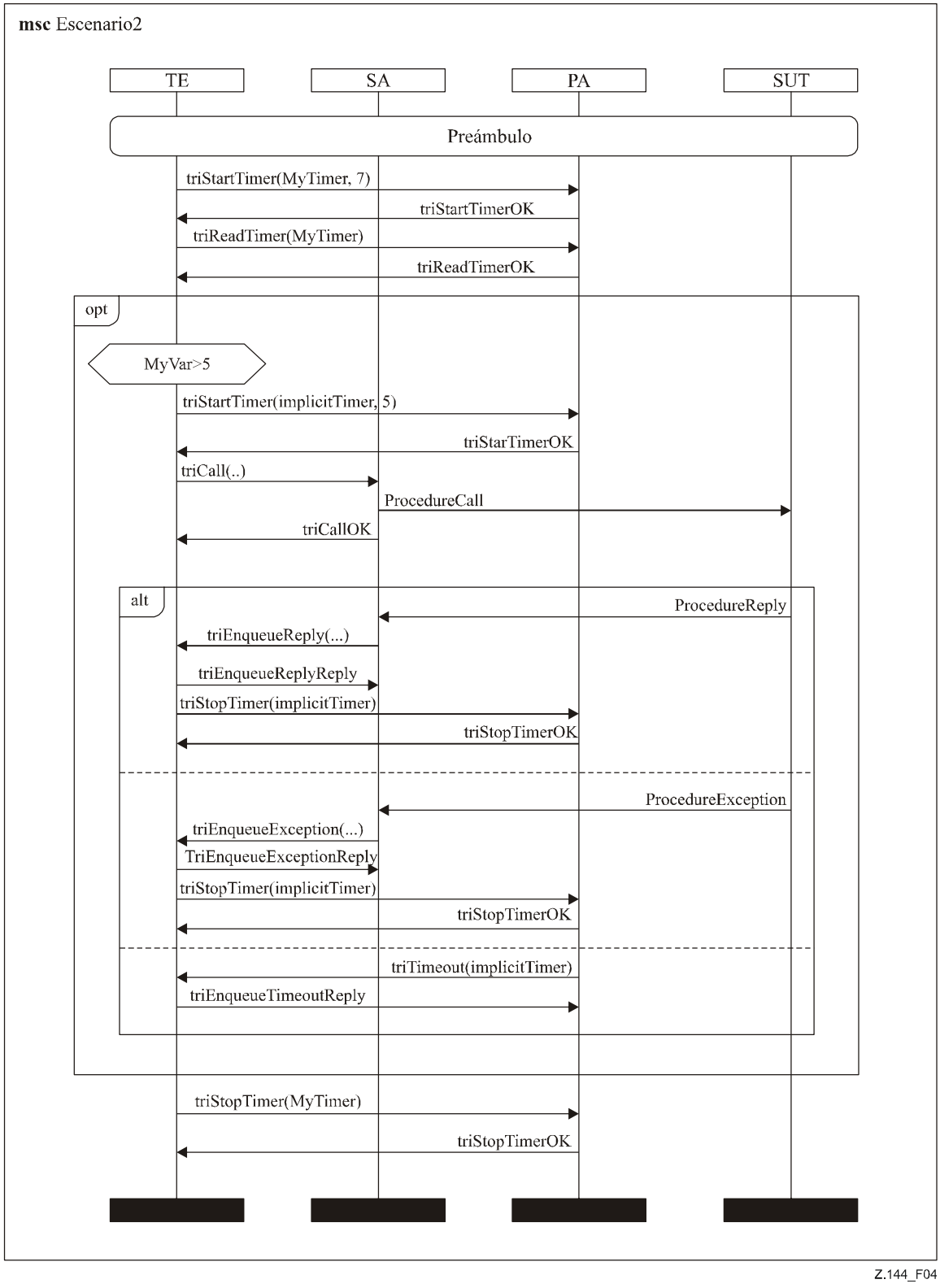


**Figura 4/Z.144 – Escenario de utilización 2**

## 8.3 Tercer escenario

En este escenario 3 se muestra la recepción de una llamada a un procedimiento y la respuesta y generación de una excepción resultantes de la recepción de esta llamada. De nuevo, se supone que `MyComponent` se comporta como un MTC. Asimismo, se supone que `FieldType1`, `FieldType2`, `p1`, y `p2` están definidos en otra parte.

### 8.3.1 Fragmento TTCN-3

```
module triScenario3
{
 signature MyProc ( in float par1, inout float par2)
    exception(MyExceptionType);

 type record MyExceptionType { FieldType1 par1, FieldType2 par2 }

 type port PortTypeProc procedure { in MyProc }

 type component MyComponent {
    port PortTypeProc MyPort;
    timer MyTimer = 3
 }

 testcase scenario3(integer x) runs on MyComponent
 {
    MyPort.start;
    MyTimer.start;
    alt
    {
     [] MyPort.getcall(MyProc:{5.0, 6.0})
     {
     MyTimer.stop;
     }
     [x>5] MyTimer.timeout
     {
     MyPort.reply(MyProc:{-, 30.0});
     }
     [x<=5] MyTimer.timeout
     {
     MyPort.raise(MyProc, MyExceptionType:{p1, p2} );
     }
    }
    MyPort.stop;
 }

 control {
    execute( scenario3(4) );
 }
}
```

### 8.3.2 Diagrama secuencial de mensajes

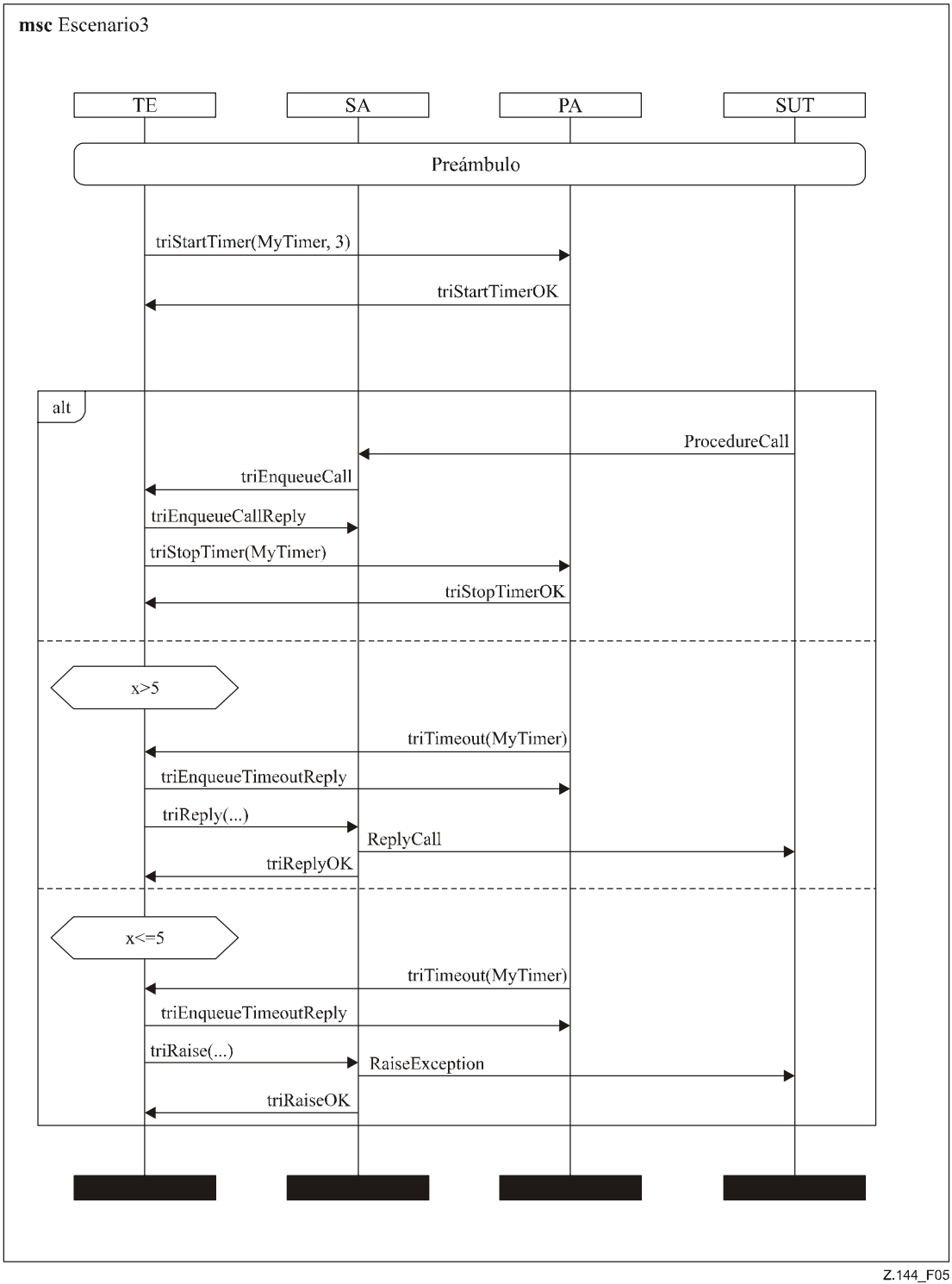


**Figura 5/Z.144 – Escenario de utilización 3**

# Anexo A (normativo)

## Resumen del IDL

En este anexo se resume la definición IDL de las operaciones TRI definidas en la cláusula 5.

```
// ***************************************************************************
// Interface definition for the TTCN-3 Runtime Interface

// ***************************************************************************

module triInterface
{

 //
 // ***************************************************************************
 // Types
 // ***************************************************************************
 //

 // Connection
 native TriPortIdType;
 typedef sequence<TriPortIdType> TriPortIdListType;
 native TriComponentIdType;
 typedef sequence<TriComponentIdType> TriComponentIdListType;

 // Communication
 native TriMessageType;
 native TriAddressType;
 typedef sequence<TriAddressType> TriAddressListType;
 native TriSignatureIdType;
 native TriParameterType;
 typedef sequence<TriParameterType> TriParameterListType;
 native TriExceptionType;

 // Timing
 native TriTimerIdType;
 native TriTimerDurationType;

 // Miscellaneous
 native TriFunctionIdType;
 native TriTestCaseIdType;
 native TriStatusType;

 //
 // ***************************************************************************
 // Interfaces
 // ***************************************************************************
 //

 //
 // ***************************************************************************
 // The communication interface (Ref: TRI-Definition: 5.5)
 // ***************************************************************************
 //
 interface triCommunication
 {

    // Reset operation

    // Ref: TRI-Definition 5.5.1
    TriStatusType triSAReset();


    // Connection handling operations

    // Ref: TRI-Definition 5.5.2.1
    TriStatusType triExecuteTestCase(in TriTestCaseIdType testCaseId,
    in TriPortIdListType tsiPortList);

    // Ref: TRI-Definition 5.5.2.2
    TriStatusType triMap(in TriPortIdType compPortId, in TriPortIdType tsiPortId);

    // Ref: TRI-Definition 5.5.2.3
    TriStatusType triUnmap(in TriPortIdType compPortId, in TriPortIdType tsiPortId);
```

```
    // Message based communication operations

    // Ref: TRI-Definition 5.5.3.1
    TriStatusType triSend(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressType SUTaddress, in TriMessageType sendMessage);
    // Ref: TRI-Definition 5.5.3.2
    TriStatusType triSendBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriMessageType sendMessage);
    // Ref: TRI-Definition 5.5.3.3
    TriStatusType triSendMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressListType SUTaddresses, in TriMessageType sendMessage);

    // Ref: TRI-Definition 5.5.3.4
    void triEnqueueMsg(in TriPortIdType tsiPortId , in TriAddressType SUTaddress,
     in TriComponentIdType componentId, in TriMessageType receivedMessage);


    // Procedure based communication operations

    // Ref: TRI-Definition 5.5.4.1
    TriStatusType triCall(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressType SUTaddress, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList);
    // Ref: TRI-Definition 5.5.4.2
    TriStatusType triCallBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriSignatureIdType signatureId,
    in TriParameterListType parameterList);
    // Ref: TRI-Definition 5.5.4.3
    TriStatusType triCallMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList);

    // Ref: TRI-Definition 5.5.4.4
    TriStatusType triReply(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressType SUTaddress, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList, in TriParameterType returnValue );
    // Ref: TRI-Definition 5.5.4.5
    TriStatusType triReplyBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriSignatureIdType signatureId,
    in TriParameterListType parameterList, in TriParameterType returnValue );
    // Ref: TRI-Definition 5.5.4.6
    TriStatusType triReplyMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList, in TriParameterType returnValue );

    // Ref: TRI-Definition 5.5.4.7
    TriStatusType triRaise(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressType SUTaddress, in TriSignatureIdType signatureId,
    in TriExceptionType exc);
    // Ref: TRI-Definition 5.5.4.8
    TriStatusType triRaiseBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriSignatureIdType signatureId,
    in TriExceptionType exc);
    // Ref: TRI-Definition 5.5.4.9
    TriStatusType triRaiseMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
    in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId,
    in TriExceptionType exc);

    // Ref: TRI-Definition 5.5.4.10
    void triEnqueueCall(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
    in TriComponentIdType componentId, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList );

    // Ref: TRI-Definition 5.5.4.11
    void triEnqueueReply(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
    in TriComponentIdType componentId, in TriSignatureIdType signatureId,
    in TriParameterListType parameterList, in TriParameterType returnValue );

    // Ref: TRI-Definition 5.5.4.12
    void triEnqueueException(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
    in TriComponentIdType componentId, in TriSignatureIdType signatureId,
    in TriExceptionType exc);


    // Miscellaneous operations

    // Ref: TRI-Definition 5.5.5.1
    TriStatusType triSUTactionInformal(in string description);

};
```

```
 //
 // ***************************************************************************
 // The platform interface (Ref: TRI-Definition: 5.6)
 // ***************************************************************************
 //
 interface triPlatform
 {

    // Reset Operation

    // Ref: TRI-Definition 5.6.1
    TriStatusType triPAReset();


    // Timer handling operations

    // Ref: TRI-Definition 5.6.2.1
    TriStatusType triStartTimer(in TriTimerIdType timerId,
    in TriTimerDurationType timerDuration);

    // Ref: TRI-Definition 5.6.2.2
    TriStatusType triStopTimer(in TriTimerIdType timerId);

    // Ref: TRI-Definition 5.6.2.3
    TriStatusType triReadTimer(in TriTimerIdType timerId,
    out TriTimerDurationType elapsedTime);

    // Ref: TRI-Definition 5.6.2.4
    TriStatusType triTimerRunning(in TriTimerIdType timerId, out boolean running);

    // Ref: TRI-Definition 5.6.2.5
    void triTimeout(in TriTimerIdType timerId);


    // Miscellaneous operations

    // Ref: TRI-Definition 5.6.3.1
    TriStatusType triExternalFunction(in TriFunctionIdType functionId,
     inout TriParameterListType parameterList,
     out TriParameterType returnValue);

 };
};
```

# BIBLIOGRAFÍA


– OMG CORBA (V2.2): *The Common Object Request Broker: Architecture and Specification*, Section 3, febrero de 1998.

– INTOOL CGI/NPL038 (V2.2): *Generic Compiler/Interpreter interface*; GCI Interface Specification, Infrastructural Tools, diciembre de 1996.

## 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 multimedios |
| Serie I | Red digital de servicios integrados |
| Serie J | Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios |
| 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