Union internationale des télécommunications

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

Z.145

(03/2006)

SÉRIE Z: LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

Techniques de description formelle – Notation de test et de commande de test

---

Notation de test et de commande de test version 3 (TTCN-3): interface de commande

Recommandation UIT-T Z.145

RECOMMANDATIONS UIT-T DE LA SÉRIE Z

LANGAGES ET ASPECTS GÉNÉRAUX LOGICIELS DES SYSTÈMES DE TÉLÉCOMMUNICATION

TECHNIQUES DE DESCRIPTION FORMELLE	
Langage de description et de spécification (SDL)	Z.100–Z.109
Application des techniques de description formelle	Z.110–Z.119
Diagrammes des séquences de messages	Z.120–Z.129
Langage étendu de définition d'objets	Z.130–Z.139
Notation de test et de commande de test	**Z.140–Z.149**
Notation de prescriptions d'utilisateur	Z.150–Z.159
LANGAGES DE PROGRAMMATION	
CHILL: le langage de haut niveau de l'UIT-T	Z.200–Z.209
LANGAGE HOMME-MACHINE	
Principes généraux	Z.300–Z.309
Syntaxe de base et procédures de dialogue	Z.310–Z.319
LHM étendu pour terminaux à écrans de visualisation	Z.320–Z.329
Spécification de l'interface homme-machine	Z.330–Z.349
Interfaces homme-machine orientées données	Z.350–Z.359
Interfaces homme-machine pour la gestion des réseaux de télécommunication	Z.360–Z.379
QUALITÉ	
Qualité des logiciels de télécommunication	Z.400–Z.409
Aspects qualité des Recommandations relatives aux protocoles	Z.450–Z.459
MÉTHODES	
Méthodes de validation et d'essai	Z.500–Z.519
INTERGICIELS	
Environnement de traitement réparti	Z.600–Z.609

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T Z.145

Notation de test et de commande de test version 3 (TTCN-3): interface de commande

Résumé

La présente Recommandation spécifie les interfaces de commande pour les implémentations de système de test en notation TTCN-3 (*version 3 de la notation de test et de commande de test*). Les interfaces de commande en notation TTCN-3 fournissent l'adaptation recommandée pour la gestion, pour le traitement des composants de test et pour le codage/décodage d'un système de test sur une plate-forme de test particulière. La présente Recommandation définit les interfaces comme étant un ensemble d'opérations indépendantes d'un langage cible.

Les interfaces sont définies de façon à être compatibles avec la Rec. UIT-T Z.140. La définition d'interface décrite dans la présente Recommandation utilise le langage de définition d'interface (IDL, *interface definition language*) de l'architecture CORBA afin de spécifier complètement l'interface TCI. Les paragraphes 8 et 9 spécifient les mappages linguistiques de la spécification abstraite vers les langages cibles Java et ANSI-C. Un résumé de la spécification d'interface fondée sur le langage IDL est présenté dans l'Annexe A.

Source

La Recommandation UIT-T Z.145 a été approuvée le 16 mars 2006 par la Commission d'études 17 (2005-2008) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas des renseignements les plus récents, il est vivement recommandé aux développeurs de consulter la base de données des brevets du TSB sous http://www.itu.int/ITU-T/ipr/.

TABLE DES MATIÈRES

Recommandation UIT-T Z.145

Notation de test et de commande de test version 3 (TTCN-3): interface de commande

1 Domaine d'application

La présente Recommandation spécifie les interfaces de commande pour les implémentations de système de test en notation TTCN-3. Les interfaces de commande en notation TTCN-3 offrent une adaptation normalisée pour la gestion, pour le traitement des composants de test et pour le codage/décodage d'un système de test sur une plate-forme de test particulière. La présente Recommandation définit les interfaces comme étant un ensemble d'opérations indépendantes d'un langage cible.

Les interfaces sont définies de façon à être compatibles avec la norme relative à la notation TTCN-3 (voir références ci-dessous). La définition d'interface utilise le langage de définition d'interface (IDL) de l'architecture CORBA afin de spécifier complètement l'interface TCI. Les paragraphes 8 et 9 spécifient les mappages linguistiques de cette spécification abstraite vers les langages cibles Java et ANSI-C. Un résumé de la spécification d'interface fondée sur le langage IDL est fournie dans l'Annexe A.

2 Références normatives

La présente Recommandation se réfère à certaines dispositions des Recommandations UIT-T et textes suivants qui, de ce fait, en sont partie intégrante. Les versions indiquées étaient en vigueur au moment de la publication de la présente Recommandation. Toute Recommandation ou tout texte étant sujet à révision, les utilisateurs de la présente Recommandation sont invités à se reporter, si possible, aux versions les plus récentes des références normatives suivantes. La liste des Recommandations de l'UIT-T en vigueur est régulièrement publiée. La référence à un document figurant dans la présente Recommandation ne donne pas à ce document, en tant que tel, le statut d'une Recommandation.

[1] Recommandation UIT-T Z.144 (2006), *Notation de test et de commande de test version 3 (TTCN-3): interface d'exécution*.

[2] Recommandation UIT-T Z.140 (2006), *Notation de test et de commande de test version 3 (TTCN-3): langage noyau*.

[3] Recommandation UIT-T Z.143 (2006), *Notation de test et de commande de test version 3 (TTCN-3): sémantique opérationnelle*.

[4] Recommandation UIT-T X.290 (1995), *Cadre général et méthodologie des tests de conformité d'interconnexion des systèmes ouverts pour les Recommandations sur les protocoles pour les applications de l'UIT-T – Concepts généraux*.

ISO/CEI 9646-1:1994, *Technologies de l'information – Interconnexion de systèmes ouverts – Cadre général et méthodologie des tests de conformité – Partie 1: Concepts généraux*.

[5] Recommandation du groupe W3C (2004), *Schéma en langage XML – Partie 0: Préliminaires*.
NOTE – Voir http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[6] Recommandation du groupe W3C (2004), *Schéma en langage XML – Partie 1: Structures*.
NOTE – Voir http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/.

[7] Recommandation du groupe W3C (2004), *Schéma en langage XML – Partie 2: Types de données*.
NOTE – Voir http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/.

3 Définitions et abréviations

3.1 Définitions

Pour les besoins de la présente Recommandation, les termes et définitions indiqués dans la Rec. UIT-T X.290 [4] s'appliquent, ainsi que les termes suivants.

3.1.1 suite de tests abstraits (ATS, *abstract test suite*): suite de tests composée de tests élémentaires abstraits

3.1.2 codec: entité de codage-décodage servant, selon le cas, à coder ou à décoder des données à envoyer et à recevoir.

3.1.3 codage/décodage (CD): entité qui administre le traitement de valeur de type y compris le codage et le décodage dans le système de test TTCN-3.

3.1.4 traitement de composant (CH, *component handling*): entité qui administre le traitement de composants de test dans le système de test TTCN-3.

3.1.5 port de communication: mécanisme abstrait facilitant la communication entre composants de test

NOTE – Un port de communication est modélisé comme une file d'attente FIFO dans le sens de la réception. Les ports peuvent être composés de messages, être composés de procédures ou être un mélange des deux.

3.1.6 composant de commande: composant qui exécute le comportement de la partie commande d'un module TTCN-3.

3.1.7 suite de tests exécutable (ETS, *executable test suite*): voir la Rec. UIT-T X.290 [4].

3.1.8 informations supplémentaires sur l'implémentation destinée au test (IXIT, *implementation extra information for testing*): voir la Rec. UIT-T X.290 [4].

3.1.9 adaptateur de plate-forme (PA, *platform adaptor*): entité qui adapte l'exécutable TTCN-3 à une plate-forme d'exécution particulière.

NOTE – L'adaptateur de plate-forme crée une unique notion de temps pour un système de test TTCN-3 et met en œuvre les deux types de temporisateur: explicite et implicite, ainsi que des fonctions externes.

3.1.10 interface réelle avec le système de test: voir la Rec. UIT-T X.290 [4].

3.1.11 système sous test (SUT, *system under test*): voir la Rec. UIT-T X.290 [4].

3.1.12 adaptateur de système sous test (SA, *SUT adaptor*): entité qui adapte les opérations de communication TTCN-3 au système SUT sur la base d'une interface abstraite avec le système de test. Cette entité implémente l'interface réelle avec le système de test.

3.1.13 notation de test et de commande de test (TTCN-3): voir la Rec. UIT-T X.290 [4].

3.1.14 test élémentaire: voir la Rec. UIT-T X.290 [4].

3.1.15 événement de test: envoi ou réception de données de test (message ou appel de procédure) à un port de communication qui fait partie de l'interface avec le système de test, ainsi qu'événement d'expiration de temporisateur.

3.1.16 gestion de test (TM): entité qui fournit une interface avec l'utilisateur ainsi que l'administration du système de test TTCN-3.

3.1.17 journalisation de test (TL, *test logging*): entité qui fournit des informations de journalisation sur l'exécution de test (y compris également les informations fournies par l'instruction de journalisation TTCN-3).

3.1.18 gestion et commande de test (TMC, *test management and control*): ensemble d'entités fournissant la gestion et commande de test; cet ensemble se compose de la gestion de test (TM), du traitement de composant (CH), de la journalisation de test (TL) et du codage/décodage (CD).

NOTE – La gestion TMC est une implémentation d'interface TCI.

3.1.19 système de test: voir la Rec. UIT-T X.290 [4].

3.1.20 interface avec le système de test (TSI, *test system interface*): composant de test qui fournit un mappage des ports disponibles dans le système (abstrait) de test TTCN-3 vers les ports offerts par un système de test réel.

3.1.21 exécutable TTCN-3 (TE, *TTCN-3 executable*): partie d'un système de test qui se rapporte à l'interprétation ou à l'exécution d'une suite ETS en notation TTCN-3.

3.1.22 interfaces de commande TTCN-3 (TCI, *TTCN-3 control interface*): ensemble des trois interfaces qui définissent l'interaction de l'exécutable TTCN-3 avec la gestion de test, avec le codage/décodage et avec le traitement de composant de test d'un système de test.

3.1.23 interface d'exécution TTCN-3 (TRI, *TTCN-3 runtime interface*): interface qui définit l'interaction de l'exécutable TTCN-3 avec le système SUT et avec les adaptateurs de plate-forme d'un système de test.

3.2 Abréviations

La présente Recommandation utilise les abréviations suivantes:

ATS suite de tests abstraits (*abstract test suite*)

CD codage/décodage

CH traitement de composant (*component handling*)

ETS suite de tests exécutable (*executable test suite*)

IDL langage de définition d'interface (*interface definition language*)

IXIT informations supplémentaires sur l'implémentation destinées au test (*implementation extra information for testing*)

MSC diagramme de séquences de messages (*message sequence chart*)

MTC composant de test principal (*main test component*)

OMG groupe de gestion par objets (*object management group*)

PA adaptateur de plate-forme (*platform adaptor*)

PTC composant de test parallèle (*parallel test component*)

SA adaptateur de système sous test (*SUT adaptor*)

SUT système sous test (*system under test*)

TC commande de test (*test control*)

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

TE (code) exécutable TTCN-3 (*TTCN-3 executable*)

TL journalisation de test (*test logging*)

TM gestion de test (*test management*)

TMC gestion et commande de test (*test management and control*)

TRI interface d'exécution TTCN-3 (*TTCN-3 runtime interface*)

TSI interface avec le système de test (*test system interface*)

TTCN-3 version 3 de la notation de test et de commande de test (*testing and test control notation version 3*)

4 Introduction

La présente Recommandation se compose de deux parties distinctes, la première décrivant la structure d'une implémentation de système de test en notation TTCN-3 et la seconde présentant la spécification des interfaces de commande en notation TTCN-3.

La première partie introduit la décomposition d'un système de test en notation TTCN-3 en quatre entités principales:

- gestion et commande de test (TMC);
- (code) exécutable TTCN-3 (TE);
- adaptateur de système sous test (SA);
- adaptateur de plate-forme (PA).

La gestion TMC se compose elle-même de trois entités: gestion de test (TM), codage/décodage (CD) et dispositif de traitement de composant de test (CH). En outre, l'interaction entre ces entités, c'est-à-dire les interfaces correspondantes, est définie.

La seconde partie de la présente Recommandation spécifie les interfaces de commande en notation TTCN-3 (TCI). Ces interfaces sont définies en termes d'opérations implémentées dans le cadre d'une même entité et appelées par d'autres entités du système de test. Pour chaque opération, la spécification d'interface définit les structures de données associées, l'effet escompté sur le système de test et toutes éventuelles contraintes sur l'usage de cette opération. Noter que ces spécifications d'interface définissent seulement des interactions entre TE et TM, entre TE et CD et entre TE et CH. Pour les interactions entre TE et SA et entre TE et PA, voir la Spécification d'interface d'exécution en notation TTCN-3 (Rec. UIT-T Z.144 [1]).

5 Conformité

Le minimum requis pour un système de test TTCN-3 conforme à l'interface TCI consiste à appliquer la spécification d'interface indiquée dans la présente Recommandation. La sémantique de la notation TTCN-3 dans le système de test doit appliquer la sémantique opérationnelle définie dans la Rec. UIT-T Z.143 [3]. En outre, un seul mappage linguistique doit être pris en charge. Par exemple, si un vendeur prend en charge le langage Java, les appels et les exécutions d'opération à l'interface TCI, qui font partie de l'exécutable TTCN-3, doivent être conformes au mappage du langage IDL vers le langage Java spécifié dans la présente Recommandation. Pour l'interface de journalisation, le mappage vers le langage XML peut être utilisé au lieu du mappage vers le langage Java ou C.

6 Structure générale d'un système de test en notation TTCN-3

Un système de test TTCN-3 peut théoriquement être considéré comme un ensemble d'entités interactives. Chaque entité met en œuvre une fonctionnalité spécifique du système de test. Ces entités:

- gèrent l'exécution de test;
- interprètent ou exécutent du code TTCN-3 compilé;
- réalisent une communication appropriée avec le système SUT;
- administrent des types, des valeurs et des composants de test;
- implémentent des fonctions externes;
- gèrent les opérations de temporisation.

6.1 Entités d'un système de test en notation TTCN-3

La structure d'une implémentation de système de test en notation TTCN-3 est illustrée dans la Figure 1.

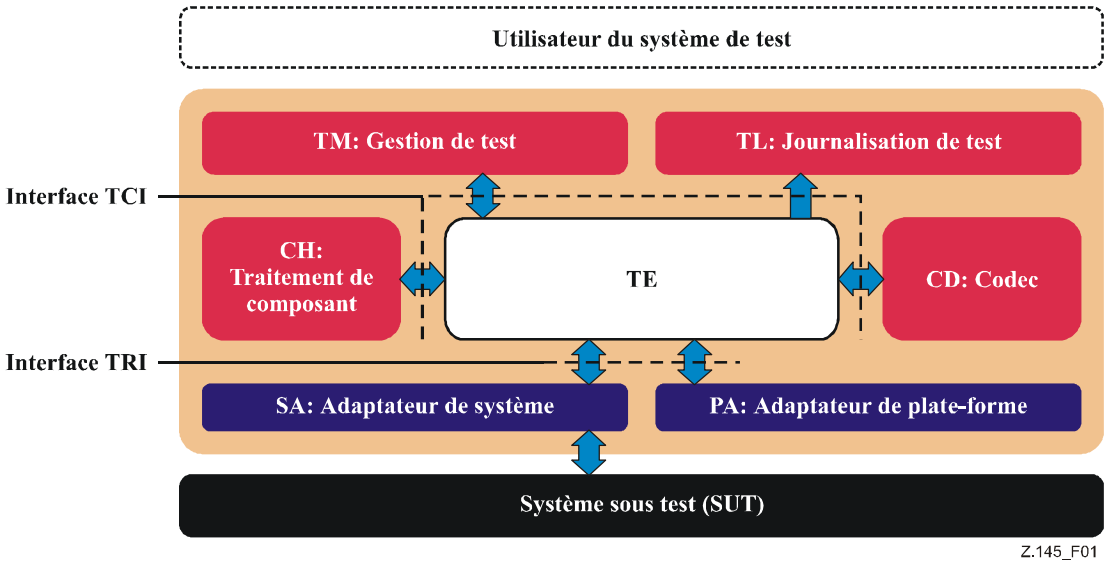

Figure 1/Z.145 – Structure générale d'un système de test en notation TTCN-3

Comme représenté dans la Figure 1, l'exécutable TTCN-3 (TE), également désigné par le terme de *suite de tests exécutable* (ETS), interprète et exécute des modules en notation TTCN-3. Divers éléments structuraux d'exécutable TE peuvent être identifiés: commande, comportement, composants, types, valeurs et files d'attente. Les éléments structuraux contenus dans l'exécutable TE représentent une fonctionnalité qui est définie à l'intérieur d'un module TTCN-3 ou par la Recommandation relative à la notation TTCN-3 (Rec. UIT-T Z.140 [2]) proprement dite. Par exemple, l'élément structurel "Commande" représente la partie commande d'un module TTCN-3, alors que l'élément structurel "Files d'attente" représente l'exigence, dans un exécutable TTCN-3, que chaque port d'un composant de test conserve sa propre file d'attente de port. Alors que le premier élément est spécifié à l'intérieur d'un module TTCN-3, le second élément est requis par la spécification TTCN-3.

Un raffinement de l'exécutable TE, représenté dans la Figure 1, est fourni afin d'aider à définir les interfaces de commande en notation TTCN-3. A l'intérieur d'une implémentation de système de test, l'exécutable TE correspondra normalement au code exécutable produit par un compilateur TTCN-3 ou par un interpréteur TTCN-3.

Le code TE peut être exécuté en mode centralisé ou en mode réparti, c'est-à-dire dans un unique dispositif de test ou dans plusieurs dispositifs de tests, selon le cas. Bien que les entités structurelles de l'exécutable TE implémentent un module TTCN-3 complet, des entités structurelles isolées pourront être réparties entre plusieurs dispositifs de test.

L'exécutable TE implémente un module TTCN-3 à un niveau abstrait. Les autres entités d'un système de test en notation TTCN-3 concrétisent ces concepts abstraits. Par exemple, le concept abstrait d'envoi d'un message ou de réception d'une fin de temporisation ne peut pas être implémenté dans l'exécutable TE. La partie restante du système de test implémente le codage du message et son envoi par des moyens physiques concrets, ou implémente le mesurage chronologique et la détermination du moment où un temporisateur est arrivé à expiration, selon le cas.

Les adaptateurs SA et PA sont, ainsi que leur interaction avec l'exécutable TE, définis dans la Rec. UIT-T Z.144 [1]. La spécification de l'interface TCI définit l'interaction entre l'exécutable TE et la gestion TMC.

L'interface de journalisation fournit des capacités de journalisation à tous les éléments de l'architecture du système de test, c'est-à-dire que l'exécutable TE, la gestion TM, le dispositif CH, le codec CD, l'adaptateur SA et l'adaptateur PA sont en mesure de journaliser des informations sur l'exécution de test au moyen de la fonction TL. La Figure 2 représente une vue plus détaillée de la fonction TL.

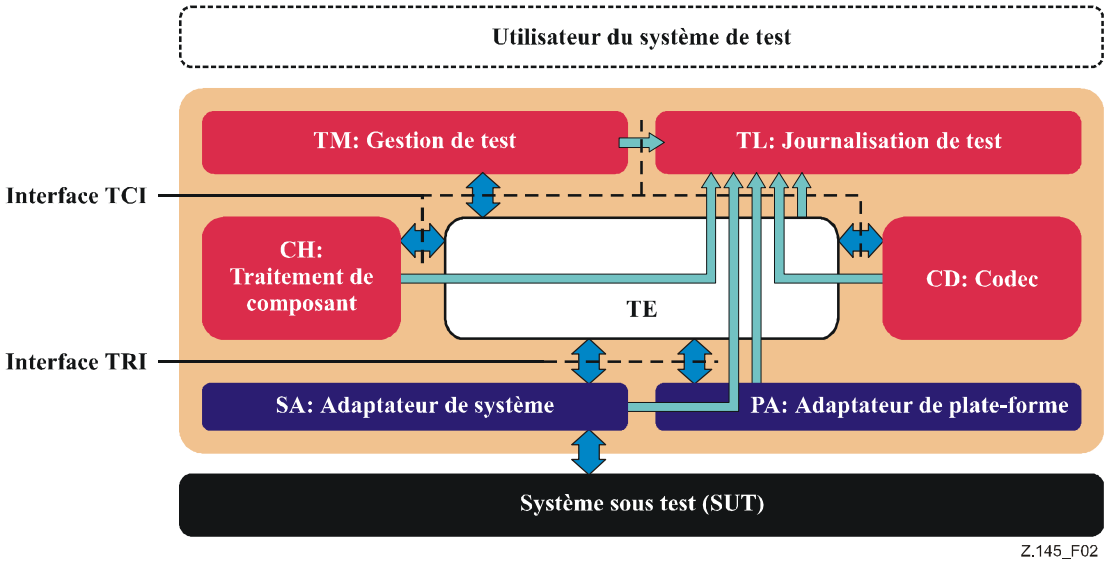

Figure 2/Z.145 – Vue détaillée de la fonction de journalisation TL

6.1.1 Gestion et commande de test (TMC)

L'entité de gestion TMC contient une fonctionnalité associée à la gestion:

- de l'exécution de test;
- des composants;
- du codage/décodage.

6.1.1.1 Gestion de test (TM)

L'entité de gestion TM est chargée de la gestion globale d'un système de test. Une fois que le système de test a été initialisé, l'exécution du test commence dans l'entité de gestion TM, qui est chargée de l'invocation appropriée des

modules en notation TTCN-3, c'est-à-dire de la propagation vers l'exécutable TE, si nécessaire, de paramètres de module tels que les informations IXIT. Normalement, cette entité implémente également une interface entre le système de test et l'utilisateur.

6.1.1.2 Codage et décodage (CD)

L'entité CD est chargée du codage et du décodage de valeurs TTCN-3 afin de constituer des chaînes binaires pouvant être envoyées au système sous test. L'exécutable TE détermine les codecs qui doivent être utilisés. Il transmet les données TTCN-3 au codeur approprié afin d'obtenir les données codées. Les données reçues sont décodées dans l'entité CD au moyen du décodeur approprié, qui convertit les données reçues en valeurs TTCN-3.

6.1.1.3 Traitement de composant (CH)

L'exécutable TE peut être réparti entre plusieurs dispositifs de test. Le dispositif CH implémente une communication entre entités réparties du système de test. L'entité CH permet de synchroniser des entités du système de test qui pourraient être réparties entre plusieurs nœuds.

NOTE 1 – Nœuds et dispositifs de test sont utilisés comme synonymes.

La structure générale d'un système de test réparti entre plusieurs nœuds est décrite dans la Figure 3.

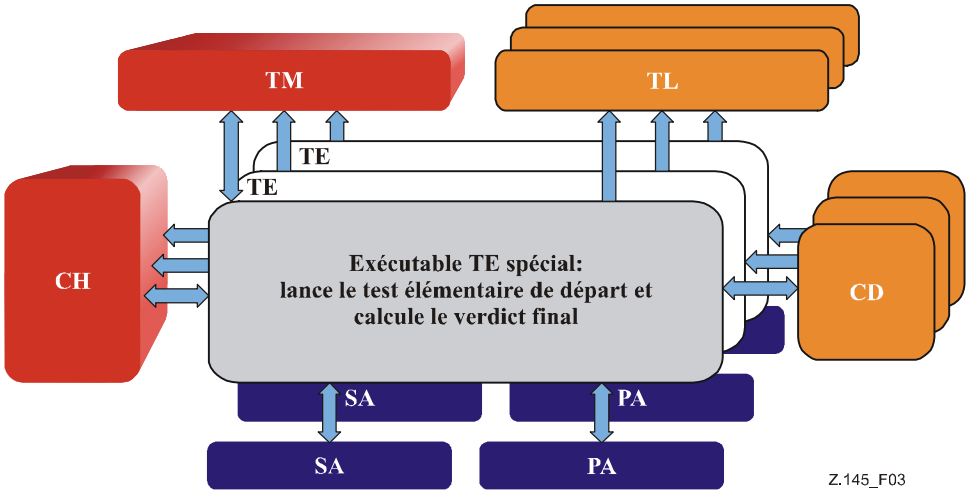

Figure 3/Z.145 – Structure générale d'un système réparti de test TTCN-3

Chaque nœud d'un système de test contient les entités TE, SA, PA, CD et TL. Les entités CH et TM facilitent la gestion de test et le traitement de composant de test entre les exécutables TE présents dans chaque nœud. L'exécutable TE qui lance un test élémentaire est un *exécutable TE spécial*. Il doit calculer le verdict final du test élémentaire. Par ailleurs, tous les exécutables TE subissent le même traitement.

NOTE 2 – Un système de test doit exécuter au plus un seul test élémentaire à un moment donné. C'est-à-dire qu'un module TTCN-3 ne peut pas exécuter plusieurs tests élémentaires en même temps.

La création des composants MTC, PTC et de commande dans les exécutables TE est contrôlée par le dispositif CH. Noter le rôle spécial du composant systémique, qui n'existe que théoriquement et non pas comme composant de test exécutant un code TE. Les ports du système, c'est-à-dire ceux du composant systémique, peuvent être répartis entre plusieurs nœuds. Par ailleurs, des composants de test situés dans différents nœuds peuvent avoir accès au même port physique du système SUT, c'est-à-dire qu'ils peuvent être mappés vers le même port de l'interface avec le système de test.

EXEMPLE: l'accès à des ports réels de système SUT distant peut être réalisé par des exécutables TE au moyen de mandataires locaux.

La communication entre composants TTCN-3 est effectuée en mode message ou en mode procédure. Le dispositif CH adapte donc la communication en mode message et en mode procédure de composants TTCN-3 à la plate-forme d'exécution particulière du système de test. Ce dispositif est informé des connexions entre les ports de communication des composants de test TTCN-3. Il propage des opérations de requête d'envoi à partir d'un composant TTCN-3 vers un autre composant TTCN-3. Le composant récepteur peut résider dans une instance différente du même exécutable TE, située dans un autre nœud. Il informe alors l'exécutable TE de tous les événements de test éventuellement reçus en les insérant dans les files d'attente de port de l'exécutable TE.

Les opérations de communication en mode procédure entre composants TTCN-3 sont également visibles par le dispositif CH, lequel doit distinguer entre les différentes sortes de communication en mode procédure (c'est-à-dire appel, réponse et exception) puis doit les propager de la façon appropriée vers l'exécutable TE où réside le composant cible. La sémantique TTCN-3 de communication en mode procédure, c'est-à-dire l'effet d'une telle opération sur l'exécution d'un composant de test TTCN-3, doit être traitée dans l'exécutable TE.

Une communication additionnelle est requise afin de mettre en œuvre la répartition de composants de test sur plusieurs nœuds. La communication de gestion de composant contient l'indication de la création de composants de test, l'indication du démarrage de l'exécution d'un composant de test, l'indication de la répartition du verdict, ainsi que l'indication de la terminaison du composant. Le dispositif CH n'implémente pas le comportement du composant TTCN-3. En revanche, il implémente la communication entre plusieurs composants implémentés par un code exécutable TE.

6.1.1.4 Journalisation de test (TL)

L'entité TL exécute la journalisation des événements de test et leur présentation à l'utilisateur du système de test. Elle fournit la journalisation des informations sur l'exécution de test, comme l'indication des composants de test qui ont été créés, armés et terminés; l'indication des données qui sont envoyées au système SUT, reçues du système SUT et mises en correspondance avec des modèles TTCN-3; l'indication des temporisateurs qui ont été armés, arrêtés ou arrivés à expiration, etc.

6.1.2 Exécutable TTCN-3 (TE)

L'entité TE exécute ou interprète un module TTCN-3. Théoriquement, l'exécutable TE peut être décomposé en six entités interactives: commande, comportement, composant, type, valeur et file d'attente. Cette décomposition structurelle de l'exécutable TE est définie dans la Rec. UIT-T Z.144 [1]. La terminologie pour les exécutables TE définis dans la Rec. UIT-T Z.144 [1] est utilisée dans la présente Recommandation.

6.1.3 Adaptateur de système sous test (SA)

L'adaptateur SA est une implémentation de l'adaptateur de système sous test (SA) défini en [1]. La terminologie des adaptateurs SA définis dans la Rec. UIT-T Z.144 [1] est utilisée dans la présente Recommandation.

6.1.4 Adaptateur de plate-forme (PA)

L'adaptateur PA est une implémentation de l'adaptateur de plate-forme (PA) défini dans la Rec. UIT-T Z.144 [1]. La terminologie des adaptateurs PA définis dans la Rec. UIT-T Z.144 [1] est utilisée dans la présente Recommandation.

6.2 Exigences d'exécution pour un système de test TTCN-3

Chaque appel d'opération à l'interface TCI doit être traité comme une opération atomique dans l'entité appelante. L'entité appelée, qui implémente une opération à l'interface TCI, doit renvoyer la commande à l'entité appelante dès que son effet recherché a été obtenu ou si l'opération ne peut pas être menée à bien. L'entité appelée ne doit pas bloquer une implémentation de communication en mode procédure.

Comme indiqué ci-dessus, aucune hypothèse n'est formulée quant à la question de savoir si le système de test TTCN-3 ou des entités individuelles sont implémentés dans un même exécutable ou processus ou s'ils sont répartis entre différents processus ou même entre différents dispositifs de test.

Une implémentation à l'interface TCI doit répondre aux exigences susmentionnées.

7 Interface de commande et opérations en notation TTCN-3

Le présent paragraphe définit un ensemble de types de données abstraites servant à représenter les données communiquées entre l'exécutable TE et la gestion TMC. Il définit également les opérations par interface TCI en fonction de leurs signatures (quand elles sont à utiliser) et définit la nature de leurs effets sur le système de test TTCN-3.

Cette définition contient également une description plus détaillée des paramètres d'entrée requis pour chaque appel d'opération à l'interface TCI, avec une valeur de retour.

7.1 Aperçu général de l'interface TCI

L'interface TCI définit l'interaction entre les entités suivantes: l'exécutable TTCN-3 (TE), le dispositif de traitement de composant (CH), la gestion de test (TM), le codec (CD), la journalisation de test (TL) d'un système de test en notation TTCN-3. Cette interface permet à l'exécutable TE:

- de gérer l'exécution du test;
- de répartir l'exécution de composants de test entre différents dispositifs de test;
- de coder et de décoder des données de test;
- de journaliser des informations sur l'exécution du test.

L'interface TCI se compose de quatre sous-interfaces:

- **interface TCI de gestion de test (TCI-TM)**: cette interface inclut toutes les opérations requises afin de gérer l'exécution du test, de fournir des paramètres de module et des constantes externes, et d'assurer la journalisation des événements de test.
- **interface TCI de traitement de composant (TCI-CH)**: cette interface se compose des opérations requises afin d'implémenter la gestion et la communication des composants de test TTCN-3 d'un système de test centralisé ou réparti. Elle inclut les opérations visant à créer, à lancer et à arrêter des composants de test, à établir des connexions entre composants TTCN-3, à gérer des composants de test et leur verdicts, et à gérer les communications en mode message et en mode procédure entre composants TTCN-3.
- **interface TCI de codage/décodage (TCI-CD)**: cette interface inclut toutes les opérations requises afin d'extraire et d'utiliser des codecs, c'est-à-dire des codeurs ou décodeurs permettant de coder les données à envoyer, définies au moyen de l'attribut de codage TTCN-3, et de décoder les données reçues.
- **interface TLI de journalisation de test (TCI-TL)**: cette interface inclut toutes les opérations requises afin d'extraire des informations sur l'exécution de test et afin de régler le niveau de détail de ces informations.

Toutes les interfaces sont bilatérales, de sorte que les parties appelante et appelée résident dans l'exécutable TE et dans la gestion TMC du système de test. Les interfaces fournies (les opérations qu'une interface offre à l'exécutable TE) et les opérations requises (celles qu'une interface a besoin d'utiliser à partir de l'exécutable TE) sont combinées dans les sous-interfaces fournie et requise de chaque interface, c'est-à-dire dans les sous-interfaces suivantes: TCI-TM fournie/TCI-TM requise, TCI-CH fournie/TCI-CH requise, TCI-CD fournie/TCI-CD requise et TCI-TL fournie/TCI-TL requise.

7.1.1 Corrélation entre notation TTCN-3 et invocations d'opération par interface TCI

Pour certaines invocations d'opération TTCN-3, il y a une corrélation directe avec une invocation d'opération à l'interface TCI, ce qui est représenté dans le Tableau 1. Certaines des opérations TTCN-3 sont en corrélation avec une paire [requête d'opération à l'interface TCI – opération à l'interface TCI] afin d'implémenter la propagation d'opérations TTCN-3 au moyen du système de test. Pour les autres invocations d'opération à l'interface TCI, il y a une corrélation indirecte étant donné que ces opérations sont requises afin d'implémenter la sémantique TTCN-3 de concepts sous-jacents.

La corrélation représentée pour les opérations de communication en notation TTCN-3 (c'est-à-dire `send`, `call`, `reply` et `raise`) n'est vérifiée que si ces opérations sont invoquées à un port de composant de test connecté à un autre port de composant de test. La corrélation pour les opérations de communication, invoquées sur des ports de composant de test mappés vers des ports d'interface avec le système de test, est définie dans la Rec. UIT-T Z.144 [1].

Tableau 1/Z.145 – Corrélation entre notation TTCN-3 et invocations d'opération par interface TCI

Nom de l'opération TTCN-3	Nom de l'opération à l'interface TCI	Nom de l'interface TCI
`send (envoi)`	`tciSendConnected` (voir Note 1)	TCI-CH Fournie
	`tciSendConnectedBC` (voir Note 2)	
	`tciSendConnectedMC` (voir Note 3)	
	`tciEnqueueMsgConnected`	TCI-CH Requise
`call (appel)`	`tciCallConnected` (voir Note 1)	TCI-CH Fournie
	`tciCallConnectedBC` (voir Note 2)	
	`tciCallConnectedMC` (voir Note 3)	
	`tciEnqueueCallConnected`	TCI-CH Requise

Tableau 1/Z.145 – Corrélation entre notation TTCN-3 et invocations d'opération par interface TCI

Nom de l'opération TTCN-3	Nom de l'opération à l'interface TCI	Nom de l'interface TCI
reply (réponse)	tciReplyConnected (voir Note 1)	TCI-CH Fournie
	tciReplyConnectedBC (voir Note 2)	
	tciReplyConnectedMC (voir Note 3)	
	tciEnqueueReplyConnected	TCI-CH Requise
raise (propagation)	tciRaiseConnected (voir Note 1)	TCI-CH Fournie
	tciRaiseConnectedBC (voir Note 2)	
	tciRaiseConnectedMC (voir Note 3)	
	tciEnqueueRaiseConnected	TCI-CH Requise
create (création)	tciCreateTestComponentReq	TCI-CH Fournie
	tciCreateTestComponent	TCI-CH Requise
start (lancement d'un composant)	tciStartTestComponentReq	TCI-CH Fournie
	tciStartTestComponent	TCI-CH Requise
stop (arrêt d'un composant)	tciStopTestComponentReq	TCI-CH Fournie
	tciStopTestComponent	TCI-CH Requise
kill (suppression)	tciKillTestComponentReq	TCI-CH Fournie
	tciKillTestComponent	TCI-CH Requise
connect (connexion)	tciConnectReq	TCI-CH Fournie
	tciConnect	TCI-CH Requise
disconnect (déconnexion)	tciDisconnectReq	TCI-CH Fournie
	tciDisconnect	TCI-CH Requise
map (mappage)	tciMapReq	TCI-CH Fournie
	tciMap	TCI-CH Requise
unmap (démappage)	tciUnmapReq	TCI-CH Fournie
	tciUnmap	TCI-CH Requise
running (en cours d'exécution)	tciTestComponentRunningReq	TCI-CH Fournie
	tciTestComponentRunning	TCI-CH Requise
alive (en vie)	tciTestComponentAliveReq	TCI-CH Fournie
	tciTestComponentAlive	TCI-CH Requise
done (effectué)	tciTestComponentDoneReq	TCI-CH Fournie
	tciTestComponentDone	TCI-CH Requise
killed (supprimé)	tciTestComponentKilledReq	TCI-CH Fournie
	tciTestComponentKilled	TCI-CH Requise
mtc (gestion mtc)	tciGetMTCReq	TCI-CH Fournie
	tciGetMTC	TCI-CH Requise
execute (exécution)	tciTestCaseExecuteReq	TCI-CH Fournie
	tciTestCaseExecute	TCI-CH Requise
log (journalisation)	tliLog	TCI-TL Fournie
NOTE 1 – Pour communication unidiffusée. NOTE 2 – Pour communication diffusée. NOTE 3 – Pour communication multidiffusée.		

7.2 Données d'interface TCI

La spécification d'interface TCI définit un ensemble de types de données abstraites qui décrivent, à un niveau très élevé, la sorte de données qui doit être transmise par une entité appelante à une entité appelée. Les types de données abstraites servent à déterminer:

- la façon dont les données TTCN-3 sont transmises d'un exécutable TE à un codeur afin de coder des représentations de valeur TTCN-3 en chaîne binaire et inversement;

- la façon dont les données transmises d'un décodeur vers l'exécutable TE doivent être décodées à partir d'une chaîne binaire afin d'obtenir sa représentation en valeur TTCN-3.

Un ensemble d'opérations est défini afin de traiter ces types de données abstraites dans le codec.

La représentation concrète de ces types de données abstraites ainsi que la définition de types de données de base comme `string` (chaîne) et `boolean` (opérateur booléen) sont définies dans les mappages linguistiques indiqués respectivement dans les § 8 et 9.

Noter que les valeurs de chaque type de données d'identificateur doivent être uniques dans une implémentation d'un système de test, le caractère unique étant défini comme étant globalement distinct à tout instant. Cela garantit que différents objets, p. ex. deux temporisateurs, sont identifiés par des identificateurs différents et que ces identificateurs ne sont pas réutilisés.

7.2.1 Types généraux de données abstraites

Les types suivants de données abstraites sont définis et servent à la définition d'opérations par interface TCI:

7.2.1.1 Gestion

`TciModuleIdType` — Une valeur de type `TciModuleIdType` est le nom d'un module TTCN-3 spécifié dans la suite ATS en notation TTCN-3. Ce type abstrait sert à la manipulation des modules.

`TciModuleParameterIdType` — Une valeur de type `TciModuleParameterIdType` est le nom qualifié d'un paramètre de module TTCN-3 spécifié dans la suite ATS en notation TTCN-3. Ce type abstrait sert à la manipulation des paramètres de module.

`TciTestCaseIdType` — Une valeur de type `TciTestCaseIdType` est le nom qualifié d'un test élémentaire TTCN-3 spécifié dans la suite ATS en notation TTCN-3. Ce type abstrait sert à la manipulation des tests élémentaires.

`TciModuleIdListType` — Une valeur de type `TciModuleIdListType` est une liste de type `TciModuleIdType`. Ce type abstrait est utilisé lors de l'extraction de la liste des modules qui sont importés par un module TTCN-3.

`TciModuleParameterType` — Une valeur de type `TciModuleParameterType` est une structure de type `TciModuleParameterIdType` assortie d'une valeur `Value`. Ce type abstrait sert à représenter le nom du paramètre et la valeur par défaut d'un paramètre de module.

`TciModuleParameterListType` — Une valeur de type `TciModuleParameterListType` est une liste de type `TciModuleParameterType`. Ce type abstrait est utilisé lors de l'extraction des paramètres d'un module TTCN-3.

`TciParameterType` — Une valeur de type `TciParameterType` contient une valeur TTCN-3 et une valeur de type `TciParameterPassingModeType` afin de représenter le mode de communication de paramètre spécifié pour le paramètre dans la suite ATS en notation TTCN-3.

`TciParameterPassingModeType` — Une valeur de type `TciParameterPassingModeType` est soit `IN`, `INOUT` ou `OUT`. Ce type abstrait est utilisé lors du démarrage d'un test élémentaire ou quand la terminaison d'un test élémentaire est indiquée.

`TciParameterListType` — Une valeur de type `TciParameterListType` est une liste de type `TciParameterType`. Ce type abstrait est utilisé lors du démarrage d'un test élémentaire ou quand la terminaison d'un test élémentaire est indiquée.

`TciParameterTypeType` — Une valeur de type `TciParameterTypeType` est une structure de type `Type` et de type `TciParameterPassingModeType`. Ce type abstrait sert à représenter le type et le mode de communication d'un paramètre de test élémentaire.

`TciParameterTypeListType` — Une valeur de type `TciParameterTypeListType` est une liste de type `TciParameterTypeType`. Ce type abstrait sert à représenter la liste des paramètres d'un test élémentaire.

`TciTestComponentKindType` — Une valeur de type `TciTestComponentKindType` est un symbole littéral de l'ensemble des sortes de composant de test TTCN-3, c'est-à-dire `MTC`, `PTC`, `PTC_ALIVE` et `CONTROL`. Ce type abstrait sert à la manipulation des composants.

`TciTypeClassType` — Une valeur de type `TciTypeClassType` est un symbole littéral de l'ensemble des classes de types en notation TTCN-3 telles que "boolean", "float", "record", etc. Ce type abstrait sert à la manipulation des valeurs.

7.2.1.2 Communication

`TciBehaviourIdType` — Une valeur de type `TciBehaviourIdType` identifie des fonctions comportementales TTCN-3.

Des types additionnels de données abstraites avec le préfixe "Tri" sont extraits de la Rec. UIT-T Z.144 [1]: `TriPortIdType`, `TriPortIdListType`, `TriComponentIdType`, `TriComponentIdListType`, `TriAddressType`, `TriAddressListType` et `TriMessageType`.

7.2.2 Données abstraites en notation TTCN-3: types et valeurs

Le présent paragraphe définit l'ensemble des types de données abstraites qui constituent la représentation des types et valeurs TTCN-3. La fonctionnalité de chaque type de données est définie par un ensemble assorti d'opérations. Les opérations qui visent ou utilisent ce type de données abstraites renvoient soit une valeur de ce type abstrait ou un type de base comme `boolean`.

Toutes les opérations ont été définies au moyen du langage de définition d'interface (IDL). Des mappages linguistiques concrets, pour les opérations sur les types de données abstraites, sont indiqués dans les § 8 et 9. Dans certains langages, l'application d'une opération sur un type de données abstraites est représentée par une communication à l'opération (soit par valeur ou par référence, selon le mappage) de la valeur concrète en tant que paramètre. D'autres langages pourraient choisir une autre méthode de référencement à la valeur concrète, p. ex. en considérant la valeur comme un objet sur lequel une méthode correspondant à l'opération est invoquée. Afin d'indiquer l'incapacité d'exécuter une certaine tâche ou afin d'indiquer l'absence d'un paramètre facultatif dans ce qui suit, la valeur distincte `null` est utilisée: elle peut être considérée comme étant une valeur réservée indiquant une valeur spéciale. Les mappages linguistiques définiront une représentation concrète de cette valeur distincte `null`.

La représentation des types et valeurs TTCN-3 abstraits se compose de deux parties:

- un type de données abstraites `Type` qui représente tous les types TTCN-3 d'un module TTCN-3;
- différents types de données abstraites qui représentent des valeurs TTCN-3, c'est-à-dire les valeurs TTCN-3 d'un type TTCN-3 indiqué. Il peut s'agir soit des valeurs de types TTCN-3 prédéfinis ou de types TTCN-3 définis par l'utilisateur.

Pour l'accès, l'évaluation et le codage des données TTCN-3, le système de test utilise le type de données abstraites `Type` et les différents types de données de valeur abstraite. Ces types de données abstraites définissent donc le niveau d'abstraction entre l'exécutable TTCN-3 (TE) et le reste du système de test au moyen des interfaces TCI.

7.2.2.1 Données abstraites en notation TTCN-3: types

Conformément à la présente Recommandation, les types TTCN-3, prédéfinis ou définis par l'utilisateur, seront représentés aux interfaces TCI au moyen du type de données abstraites `Type`.

Pour le type de données abstraites `Type`, un ensemble d'opérations est défini:

- afin de faire référence à des types de données TTCN-3 prédéfinis ou définis par l'utilisateur;
- afin de créer et de maintenir des valeurs TTCN-3.

Les opérations suivantes sont définies pour le type de données abstraites `Type`:

`TciModuleIdType getDefiningModule()` — Renvoie l'identificateur du module dans lequel le type est défini. Renvoie la valeur distincte `null` si le type est un type de base TTCN-3, p. ex. un booléen, un entier, etc.

`TString getName()` — Renvoie le nom du type comme défini dans le module TTCN-3.

`TciTypeClassType getTypeClass()` — Renvoie la classe du type considéré. Une valeur de type `TciTypeClassType` peut avoir une des constantes suivantes: `ADDRESS`, `ANYTYPE`, `BITSTRING`, `BOOLEAN`, `CHARSTRING`, `COMPONENT`, `ENUMERATED`, `FLOAT`, `HEXSTRING`, `INTEGER`, `OBJID`, `OCTETSTRING`, `RECORD`, `RECORD_OF`, `SET`, `SET_OF`, `UNION`, `UNIVERSAL_CHARSTRING`, `VERDICT`.

`Value newInstance()` — Renvoie une valeur récemment créée du type indiqué. Cette valeur initiale de la valeur créée est indéfinie.

`TString getTypeEncoding()` — Renvoie l'attribut de codage de type défini dans le module TTCN-3.

`TString getTypeEncodingVariant()` — Cette opération renvoie l'éventuel attribut de variante du codage de la valeur comme défini dans la notation TTCN-3. Si aucun attribut de variante de codage n'est défini, la valeur distincte `null` est renvoyée.

`TStringseq getTypeExtension()` — Renvoie l'attribut d'extension de type comme défini dans le module TTCN-3.

7.2.2.2 Valeurs abstraites TTCN-3

Conformément à la présente Recommandation, les valeurs TTCN-3 sont représentées aux interfaces TCI au moyen de nombreux types de données abstraites.

La Figure 4 présente la hiérarchie entre les types de données abstraites pour les valeurs TTCN-3 (en abrégé: *valeurs abstraites*).

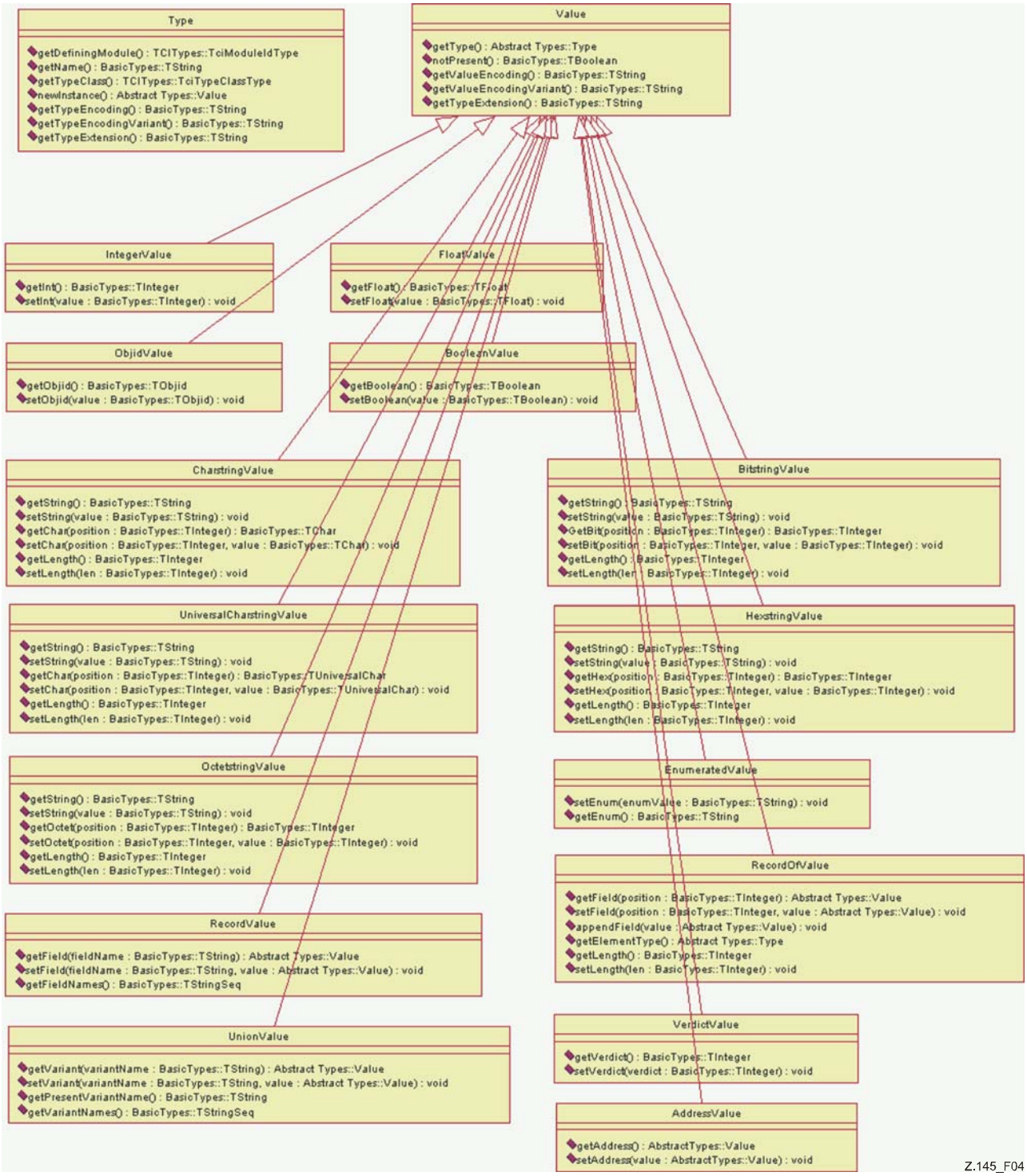

Figure 4/Z.145 – Hiérarchie de valeur abstraites

Comme représenté dans la Figure 4, toutes les valeurs abstraites TTCN-3 partagent le même type de données abstraites de base: `Value`. Toutes les opérations définies pour ce type de données de base commun sont également définies implicitement pour les types de valeur abstraite qui en sont dérivés.

7.2.2.2.1 Le type de données abstraites `Value`

Les opérations suivantes sont définies pour le type de base de données abstraites `Value`. Les représentations concrètes de ces opérations sont définies dans les paragraphes relatifs aux mappages linguistiques correspondants:

`Type getType()`	Renvoie le type de la valeur spécifiée.
`TBoolean notPresent()`	Renvoie la valeur `true` si la valeur spécifiée est `omit`, sinon renvoie la valeur `false`.
`TString getValueEncoding()`	Renvoie l'attribut de codage de valeur défini dans la notation TTCN-3, s'il existe. Si aucun attribut de codage n'est défini, la valeur distincte `null` est renvoyée.
`TString getValueEncodingVariant()`	Renvoie l'attribut de variante de codage de valeur défini dans la notation TTCN-3, s'il existe. Si aucun attribut de variante de codage n'est défini, la valeur distincte `null` est renvoyée.

7.2.2.2.2 Le type de données abstraites `IntegerValue`

Le type de données abstraites `IntegerValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs d'entier (`integer`) en notation TTCN-3.

Les opérations suivantes sont définies pour le type de données abstraites `IntegerValue`:

`TInteger getInt()`	Renvoie la valeur d'entier de cet entier TTCN-3.
`void setInt(in Value TInteger)`	Met ce type `IntegerValue` à la valeur `value`.

7.2.2.2.3 Le type de données abstraites `FloatValue`

Le type de données abstraites `FloatValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 en virgule flottante (`float`).

Les opérations suivantes sont définies pour le type de données abstraites `FloatValue`:

`TFloat getFloat()`	Renvoie la valeur en virgule flottante de cette valeur en virgule flottante TTCN-3.
`void setFloat(in Value TFloat)`	Met ce type `FloatValue` à la valeur `value`.

7.2.2.2.4 Le type de données abstraites `BooleanValue`

Le type de données abstraites `BooleanValue est` fondé sur le type de données abstraites `Value`. Il représente des valeurs booléennes TTCN-3 (`boolean`).

Les opérations suivantes sont définies pour le type de données abstraites `BooleanValue`:

`TBoolean getBoolean ()`	Renvoie la valeur booléenne de la valeur TTCN-3 `boolean`.
`void setBoolean(in Value TBoolean)`	Met cette valeur booléenne à la valeur `value`.

7.2.2.2.5 Le type de données abstraites `ObjidValue`

Le type de données abstraites `ObjidValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 d'identification d'objet (`objid`).

Les opérations suivantes sont définies pour le type de données abstraites `ObjidValue`:

`TObjid getObjid()`	Renvoie la valeur d'identification d'objet du type TTCN-3 `objid`.
`void setObjid(in Value TObjid)`	Met ce type `ObjidValue` à la valeur `value`.

7.2.2.2.6 Le type de données abstraites `CharstringValue`

Le type de données abstraites `CharstringValue est` fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 `charstring`. `TChar` est un caractère d'une valeur de chaîne de caractères.

Les opérations suivantes sont définies pour le type de données abstraites `CharstringValue`:

`TString getString()`	Renvoie la valeur en chaîne du type TTCN-3 `charstring`. La représentation textuelle de la chaîne de caractères vide TTCN-3 `charstring` est `''`, alors que sa longueur est zéro.
`void setString(in Value TString)`	Met ce type `CharstringValue` à la valeur `value`.
`TChar getChar(in position TInteger)`	Renvoie la valeur de caractère de la chaîne de caractères TTCN-3 à la position `position`. La position 0 indique le premier caractère de la chaîne de caractères TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`.
`void setChar(in position TInteger, in TChar)`	Cette opération met le caractère à sa valeur de position. Les valeurs valides pour la position vont de `0` à `length - 1`.
`TInteger getLength()`	Renvoie la longueur de ce type `CharstringValue` en nombre de caractères; renvoie zéro si la valeur de ce type `CharstringValue` est `omit`.
`void setLength(in TInteger len)`	L'opération `setLength` réinitialise d'abord ce type `CharstringValue` à sa valeur initiale, puis met à `len` la longueur de ce type `CharstringValue`, exprimée en nombre de caractères.

7.2.2.2.7 Le type de données abstraites `UniversalCharstringValue`

Le type de données abstraites `UniversalCharstringValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 de type chaîne de caractères universels (`universal charstring`). `TUniversalChar` est un caractère d'une valeur de chaîne de caractères universels.

Les opérations suivantes sont définies pour le type de données abstraites `UniversalCharstringValue`:

`TString getString()`	Renvoie la représentation textuelle de ce type `UniversalCharstringValue`, comme défini dans la notation TTCN-3.
`void setString(in Value TString)`	Règle la valeur de ce type `UniversalCharstringValue` conformément à la représentation textuelle définie par `Value`.
`TUniversalChar getChar(in position TInteger)`	Renvoie la valeur de caractère universel du type TTCN-3 `universal charstring` à la position `position`. La position 0 indique le premier caractère `TUniversalChar` de la chaîne TTCN-3 `universal charstring`. Les valeurs valides pour la position vont de `0` à `length - 1`.
`void setChar(in position TInteger, in TUniversalChar)`	Met le caractère universel à la position indiquée par la valeur value. Les valeurs valides pour la position vont de `0` à `length - 1`.
`TInteger getLength()`	Renvoie la longueur de ce CharstringValue universels en caractères universels, zéro si la valeur de chaîne de caractères universels est `omit`.
`void setLength(in TInteger len)`	L'opération setLength réinitialise d'abord ce type `UniversalCharstringValue` à sa valeur initiale puis règle à `len` la longueur de ce type `UniversalCharstringValue`, exprimée en nombre de caractères universels.

7.2.2.2.8 Le type de données abstraites `BitstringValue`

Le type de données abstraites `BitstringValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 de chaîne binaire `(bitstring)`:

les opérations suivantes sont définies pour le type de données abstraites `BitstringValue`.

`TString getString()`	Renvoie la représentation textuelle de ce type `BitstringValue`, comme défini dans la notation TTCN-3. P. ex., la représentation textuelle de `0101` est `"0101"B`. La représentation textuelle de la chaîne binaire vide en notation TTCN-3 est `""B`, alors que sa longueur est zéro.
`void setString(in Value TString)`	Règle la valeur de ce type `BitstringValue` conformément à la représentation textuelle définie par `Value`. P. ex., la valeur de ce type `BitstringValue` est `0101` si la représentation textuelle contenue dans la valeur `value` est `"0101"B`.
`TChar getBit(in position TInteger)`	Renvoie la valeur `(0 | 1)` à la position de cette chaîne binaire TTCN-3 sous forme d'un caractère. La position 0 indique le premier bit de la chaîne binaire TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`.
`void setBit(in TInteger position, in TInteger)`	Met le bit à la position indiquée par la valeur `(0 | 1)`. La position 0 indique le premier bit dans ce type `BitstringValue`. Les valeurs valides pour la position vont de `0` à `length - 1`.
`TInteger getLength()`	Renvoie la longueur de ce type `BitstringValue` en bits, zéro si la valeur de ce type `BitstringValue` est `omit`.
`void setLength(in TInteger len)`	L'opération `setLength` réinitialise d'abord ce type `BitstringValue` à sa valeur initiale puis met la longueur, exprimée en bits, de ce type `BitstringValue` à `len`.

7.2.2.2.9 Le type de données abstraites `OctetStringValue`

Le type de données abstraites `OctetstringValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs TTCN-3 de chaîne d'octets.

Les opérations suivantes sont définies pour le type de données abstraites `OctetstringValue`:

`TString getString()`	Renvoie la représentation textuelle de ce type `OctetstringValue`, comme défini dans la notation TTCN-3. Par exemple, la représentation textuelle de `0xCAFFEE` est `"CAFFEE"O`. La représentation textuelle de la chaîne d'octets TTCN-3 vide `octetstring` est `""O`, alors que sa longueur est zéro.
`void setString(in Value TString)`	Règle la valeur de ce type `OctetstringValue` conformément à la représentation textuelle définie par `Value`. P. ex., la valeur de ce type `OctetstringValue est 0xCAFFEE si la représentation textuelle` contenue dans la valeur value est `"CAFFEE"O`.
`TChar getOctet(in TInteger position)`	Renvoie la valeur `(0.255)` à la `position` de cette chaîne d'octets TTCN-3. La position 0 indique le premier octet de la chaîne d'octets TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`.
`void setOctet(in TInteger position, in TInteger)`	Met l'octet à la position indiquée par la valeur `(0..255)`. La position 0 indique le premier octet dans la chaîne d'octets. Les valeurs valides pour la position vont de `0` à `length - 1`.
`TInteger getLength()`	Renvoie la longueur de ce type `OctetstringValue` en octets, zéro si la valeur de ce type `OctetstringValue` est `omit`.
`void setLength(in TInteger len)`	L'opération `setLength` réinitialise d'abord ce type `OctetstringValue` à sa valeur initiale puis met la longueur -en octets- de ce type `OctetstringValue` à `len`.

7.2.2.2.10 Le type de données abstraites `HexstringValue`

Le type de données abstraites `HexstringValue` est fondé sur le type de données abstraites `Value`. Il représente des valeurs de chaîne hexadécimale TTCN-3.

Les opérations suivantes sont définies pour le type de données abstraites `HexstringValue`:

`TString getString()` — Renvoie la représentation textuelle de ce type `HexstringValue`, comme défini dans la notation TTCN-3. P. ex., la représentation textuelle de `0xAFFEE` est `"AFFEE"H`. La représentation textuelle de la chaîne hexadécimale TTCN-3 vide est `""H`, alors que sa longueur est zéro.

`void setString(in Value TString)` — Règle la valeur de ce type `HexstringValue` conformément à la représentation textuelle définie par la valeur value. P. ex., la valeur de ce type `HexstringValue` est `0xAFFEE` si la représentation textuelle contenue dans la valeur `value` est `"AFFEE"H`.

`TChar getHex(in TInteger position)` — Renvoie la valeur `(0.15)` à la position de cette chaîne hexadécimale TTCN-3. La position 0 indique les premiers caractères hexadécimaux de la chaîne hexadécimale TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`.

`void setHex(in TInteger position , in TInteger)` — Met le caractère hexadécimal à la position indiquée par la valeur `(0..15)`. La position 0 indique le premier octet dans la chaîne hexadécimale. Les valeurs valides pour la position vont de `0` à `length - 1`.

`TInteger getLength()` — Renvoie la longueur de ce type `HexstringValue` en octets, zéro si la valeur de ce type `HexstringValue` est `omit`.

`void setLength(in TInteger len)` — L'opération `setLength` réinitialise d'abord ce type `HexstringValue` à sa valeur initiale puis met la longueur de ce type `HexstringValue` en caractères hexadécimaux à `len`.

7.2.2.2.11 Le type de données abstraites `RecordValue`

Le type de données abstraites `RecordValue` est fondé sur le type de données abstraites `Value`. Il spécifie comment rechercher et mettre à jour le type enregistrement TTCN-3 (`record`). Le même type de données abstraites s'applique aux valeurs dont la classe de type est `SET`. La distinction entre `record` et `set` n'est pertinente qu'au moment d'une mise en correspondance.

Les opérations suivantes sont définies pour le type de données abstraites `RecordValue`:

`Value getField(in TString fieldName)` — Renvoie la valeur du champ nommé `fieldName`. La valeur de retour est le type de base abstrait commun `Value`, étant donné qu'un champ d'enregistrement peut être de tout type défini dans la notation TTCN-3. Si le champ ne peut pas être obtenu à partir de l'enregistrement, la valeur distincte `null` est renvoyée.

`void setField(in TString fieldName, in Value value)` — Met le champ nommé `fieldName` de l'enregistrement à la valeur `value`. Aucune hypothèse ne doit être faite sur la façon dont un champ est mémorisé dans un enregistrement. Une implémentation interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur est copiée. L'on devrait donc partir du principe que les modifications subséquentes de la valeur `value` ne seront pas prises en considération dans l'enregistrement.

`TStringseq getFieldNames()` — Renvoie une séquence de chaîne de noms de champ, et une séquence vide si l'enregistrement ne contient aucun champ.

7.2.2.2.12 Le type de données abstraites `RecordOfValue`

Le type de données abstraites `RecordOfValue` est fondé sur le type de données abstraites `Value`. Il spécifie comment rechercher et mettre à jour des éléments dans les types TTCN-3 `record of`. Le même type de données abstraites s'applique aux valeurs dont la classe de type est `SET_OF`. La distinction entre `record of` et `set of` n'est pertinente qu'au moment d'une mise en correspondance.

Les opérations suivantes sont définies pour le type de données abstraites `RecordOfValue`:

`Value getField(in TInteger position)`
Renvoie la valeur du type `record of` à la position `position` si le type `position` est compris entre zéro et `length - 1`, sinon renvoie la valeur distincte `null`. La valeur de retour est le type de base abstrait commun `Value` étant donné que le type `record of` peut avoir des champs de tout type défini dans la notation TTCN-3.

`void setField(in TInteger position , in Value value)`
Met le champ à la position `position` indiquée par la valeur `value`. Si `position` est supérieur à `(length -1)`, le type `record of` est étendu de façon à avoir la longueur `(position + 1)`. Les éléments de type `record of`, compris entre la position originale située à `length` et la position `position - 1`, sont réglés à `omit`. Aucune hypothèse ne doit être faite sur la façon dont un champ est mémorisé dans un type `record of`. Une implémentation interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur est copiée. L'on devrait donc supposer que les modifications subséquentes de la valeur `value` ne seront pas reflétées dans le type `record of`.

`void appendField(in Value value)`
Ajoute la valeur à la fin du type `record of`, c'est-à-dire à la position `length`. Aucune hypothèse ne doit être faite sur la façon dont un champ est mémorisé dans un type `record of`. Une implémentation interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur est copiée. L'on devrait donc supposer que les modifications subséquentes de la valeur `value` ne seront pas reflétées dans le type `record of`.

`Type getElementType()`
Renvoie le type `Type` des éléments du type `record of` considéré.

`TInteger getLength()`
Renvoie la longueur réelle de la valeur du type `record of`, et renvoie zéro si la valeur du type `record of` est `omit`.

`void setLength(in TInteger len)`
Met la longueur du type `record of` à la valeur `len`. Si `len` est supérieur à la longueur originale, les éléments récemment créés ont la valeur `omit`. Si `len` est inférieur ou égal à la longueur originale, cette opération est ignorée.

7.2.2.2.13 Le type de données abstraites `UnionValue`

Le type de données abstraites `UnionValue` est fondé sur le type de données abstraites `Value`. Il spécifie comment rechercher et mettre à jour des variantes d'un type TTCN-3 `union`. Le type TTCN-3 `anytype` est représenté par une valeur `UnionValue` où la classe du type obtenu par `getType()` est `ANYTYPE`. Pour les détails sur les classes de types, voir le § 7.2.2.1.

Les opérations suivantes sont définies pour le type de données abstraites `UnionValue`:

`Value getVariant(in TString variantName)`
Renvoie la valeur du type de réunion TTCN-3 `variantName` si `variantName` est égal au résultat de l'opération `getPresentVariantName`; sinon renvoie la valeur distincte `null`. Le type `variantName` indique le nom de la variante de réunion comme défini dans la notation TTCN-3.

`void setVariant(in TString variantName , in Value value)`
Met le nom `variantName` de la réunion à la valeur `value`. Si `variantName` n'est pas défini pour cette réunion, cette opération est ignorée. Si une autre variante a été sélectionnée, la nouvelle variante est sélectionnée à la place de la première.

`TString getPresentVariantName()`
Renvoie une chaîne représentant le nom de variante actuellement sélectionné dans la réunion TTCN-3 indiquée. La valeur distincte `null` est renvoyée si aucune variante n'est sélectionnée.

`TStringSeq getVariantNames()` — Renvoie une séquence de chaîne de noms de variante, et renvoie la valeur distincte `null` si la réunion ne contient aucun champ. Si la valeur `UnionValue` représente le type TTCN-3 `anytype`, c'est-à-dire si la classe du type obtenu par l'opération getType() est ANYTYPE, tous les types TTCN-3 prédéfinis et définis par l'utilisateur sont renvoyés.

7.2.2.2.14 Le type de données abstraites `EnumeratedValue`

Le type de données abstraites `EnumeratedValue` est fondé sur le type de données abstraites `Value`. Il spécifie comment le type TTCN-3 `enumerated` peut être mis à jour et recherché.

Les opérations suivantes sont définies pour le type de données abstraites `EnumeratedValue`:

`TString getEnum()` — Renvoie l'identificateur de chaîne de ce type `EnumeratedValue`. Cet identificateur est égal à l'identificateur indiqué dans la spécification TTCN-3.

`void setEnum(in TString enumValue)` — Met le type enum à la valeur `enumValue`. Si la valeur `enumValue` n'est pas une valeur autorisée pour cette énumération, l'opération est ignorée.

7.2.2.2.15 Le type de données abstraites `VerdictValue`

Le type de données abstraites `VerdictValue` est fondé sur le type de données abstraites `Value`. Il spécifie comment le type TTCN-3 `verdict` peut être mis à jour et recherché.

Les opérations suivantes sont définies pour le type de données abstraites `VerdictValue`:

`TInteger getVerdict()` — Renvoie la valeur d'entier pour cette valeur `VerdictValue`. Cet entier est une des constantes suivantes: `ERROR`, `FAIL`, `INCONC`, `NONE`, `PASS`.

`void setVerdict(in TInteger verdict)` — Met ce type `VerdictValue` à la valeur `verdict`. Noter qu'une valeur `VerdictValue` peut être attribuée à l'un quelconque des verdicts susmentionnés à un moment quelconque. Le type `VerdictValue` n'exécute aucun calcul de verdict comme défini dans la notation TTCN-3. Par exemple, il est permis de régler le type `VerdictValue` d'abord à `ERROR` puis à `PASS`.

7.2.2.2.16 Le type de données abstraites `AddressValue`

Les opérations suivantes sont définies pour le type de base de données abstraites `AddressValue`. Les représentations concrètes de ces opérations sont définies dans les paragraphes relatifs aux mappages linguistiques correspondants:

`Value getAddress()` — Renvoie la valeur d'adresse, qui ne sera plus de la classe du type `ADDRESS` mais plutôt celle du type réel servant à l'adressage.

`void setAddress(in Value value)` — Met cette valeur d'adresse à la valeur `value`.

7.2.3 Types de journalisation abstraite

7.2.3.1 Le type de données abstraites `TciValueTemplate`

Les opérations suivantes sont définies pour le type de données abstraites `TciValueTemplate`. Les représentations concrètes de ces opérations sont définies dans les paragraphes relatifs aux mappages linguistiques correspondants:

`boolean isOmit()` — Renvoie la valeur `true` si le modèle est de type omit.

`boolean isAny()` — Renvoie la valeur `true` e si le modèle est de type any.

`boolean isAnyOrOmit()` — Renvoie la valeur `true` si le modèle est de type any ou omit.

`TString getTemplateDef()` — Renvoie la définition de ce modèle.

7.2.3.2 Le type de données abstraites `TciNonValueTemplate`

Les opérations suivantes sont définies pour le type de données abstraites `TciNonValueTemplate`. Les représentations concrètes de ces opérations sont définies dans les paragraphes relatifs aux mappages linguistiques correspondants:

`boolean isAny()`	Renvoie la valeur `true` si le modèle est de type `any`.
`boolean isAll()`	Renvoie la valeur `true` si le modèle est de type `all`.
`TString getTemplateDef()`	Renvoie la définition de ce modèle.

7.2.3.3 La liste des valeurs et Mismatch Types

Les types suivants de données abstraites sont définis et servant à la journalisation de différences entre valeurs et modèles:

`TciValueList`	Une valeur `TciValueList` est une liste de valeurs.
`TciValueDifference`	Une valeur `TciValueDifference` est une structure contenant une valeur, une modèle et une description pour la raison de cette différence.
`TciValueDifferenceList`	Une valeur `TciValueDifferenceList` est une séquence de différences de valeur.

7.3 Opérations par interface TCI

Le présent paragraphe spécifie les opérations qu'un exécutable TTCN-3 doit fournir à un système de test (*opérations requises*) et la fonctionnalité qui doit être fournie par le système de test à l'exécutable TTCN-3 (*opérations fournies*).

Les termes "requises" et "fournies" reflètent le fait que la présente Recommandation définit les exigences applicables à un exécutable TTCN-3 du point de vue d'un utilisateur. Celui-ci "exige" d'un exécutable TTCN-3 une certaine fonctionnalité afin de construire un système de test complètement fondé sur la notation TTCN-3. Afin de remplir sa tâche, l'exécutable TTCN-3 doit informer l'utilisateur de l'existence de certains événements dans lesquels l'utilisateur doit "fournir" cette possibilité à l'exécutable TTCN-3.

Toutes les définitions d'opération figurant dans le présent paragraphe sont décrites au moyen du langage de définition d'interface (IDL). Des mappages linguistiques concrets sont définis dans les § 8 et 9. L'Annexe B présente un mappage de remplacement vers le langage XML pour l'interface de journalisation.

Pour chaque appel d'opération à l'interface TCI, tous les Paramètres in, *inout* et *out* énumérés dans la définition d'opération particulière sont obligatoires. La valeur d'un paramètre *in* est spécifiée par l'entité appelante, laquelle se rapporte au sens de l'appel. Pour les opérations sur une interface *requise*, l'entité appelante est le système de test alors que l'entité appelée est l'exécutable TTCN-3. Pour les opérations sur une interface *fournie*, l'entité appelante est l'exécutable TTCN-3 alors que le système de test est l'entité appelée.

De même, la valeur d'un paramètre *out* est spécifiée par l'entité appelée. Dans le cas d'un paramètre *inout*, une valeur est d'abord spécifiée par l'entité appelante mais peut être remplacée par une nouvelle valeur par l'entité appelée. Noter que, bien que la notation TTCN-3 utilise également les Paramètres in, *inout* et *out* pour les définitions de signature, les notations utilisées dans la spécification de l'interface TCI en langage IDL ne sont pas associés à celles qui sont utilisées dans une spécification en notation TTCN-3.

Les appels d'opération devraient utiliser une valeur réservée afin d'indiquer l'absence de paramètres. Les valeurs réservées pour ces types sont définies dans chaque mappage linguistique et seront par la suite désignées par la valeur `null`.

Par ailleurs, la valeur `null` servira également à indiquer l'incapacité d'exécuter une certaine tâche.

Comme le présent paragraphe spécifie seulement des interfaces et ne suggère pas de mises en œuvre concrètes quant à la façon d'exécuter la fonctionnalité spécifiée, le terme *entité* servira à identifier la partie d'une implémentation d'un système de test qui réalise cette interface et qui exécute la fonctionnalité demandée. Par exemple, l'entité appelante contenue dans l'opération TciSendConnected est l'exécutable TE, c'est-à-dire la partie d'une implémentation d'un système de test qui fournit la fonctionnalité d'exécutable TE.

Toutes les fonctions contenues dans l'interface sont décrites au moyen du modèle ci-dessous. Les descriptions qui ne sont pas applicables à certaines opérations sont supprimées.

Signature	`Signature en langage IDL`
Paramètres in	Description des données transmises comme paramètres à l'opération, de l'entité appelante à l'entité appelée
Paramètres out	Description des données transmises comme paramètres à l'opération, de l'entité appelée à l'entité appelante
Paramètres inout	Description des données transmises comme paramètres à l'opération, de l'entité appelante à l'entité appelée et de l'entité appelée en retour à l'entité appelante
Valeur renvoyée	Description des données renvoyées par l'opération à l'entité appelante
Contrainte	Description des contraintes éventuelles quand l'opération peut être appelée
Effet	Comportement requis de l'entité appelée avant que l'opération puisse renvoyer une valeur

7.3.1 L'interface TCI-TM

L'interface TCI de gestion de test (TCI-TM) décrit les opérations qu'un exécutable TTCN-3 est tenu d'implémenter et les opérations qu'une implémentation de gestion de test doit fournir à l'exécutable TE (Figure 5).

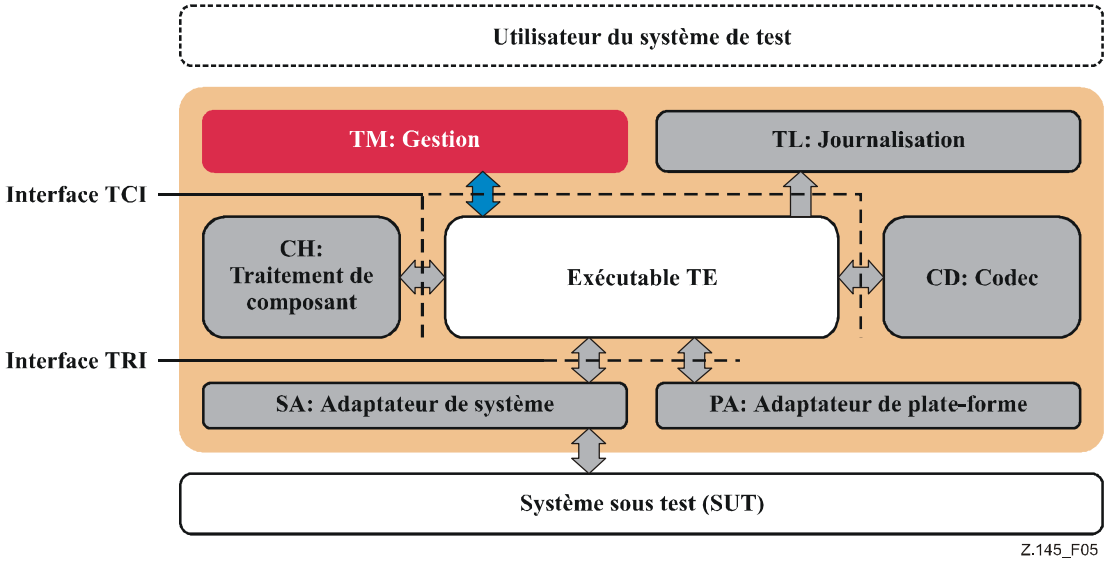

Figure 5/Z.145 – L'interface TCI-TM

Une implémentation de gestion de test fournit la gestion globale de test à l'utilisateur du système de test. Elle requiert de l'exécutable TE la présence d'opérations afin de lancer et d'arrêter l'exécution de test d'un module TTCN-3 ou de certains tests élémentaires d'un module TTCN-3. Inversement, elle fournit des opérations à l'exécutable TE afin de résoudre un paramètre de module pendant l'exécution et afin d'indiquer la terminaison de l'exécution.

Le paragraphe 10 illustre l'usage et l'ordonnancement séquentiel d'appels d'opération par l'exécutable TE ou par la gestion de test.

7.3.1.1 Sous-interface TCI-TM requise

Le présent paragraphe spécifie les opérations que la gestion TM exige de l'exécutable TE. En plus des opérations spécifiées dans le présent paragraphe, une gestion de test exige les opérations requises à l'interface TCI-CD.

7.3.1.1.1 tciRootModule

Signature	void `tciRootModule(in TciModuleIdType moduleName)`	
Paramètres in	`moduleName`	Le nom `moduleName` indique les identificateurs de module comme défini dans la notation TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Ne doit être utilisé que si ni la partie commande ni un test élémentaire n'est actuellement en cours d'exécution	
Effet	`tciRootModule` sélectionne le module indiqué pour exécution au moyen d'un appel subséquent utilisant `tciStartTestCase` ou `tciStartControl`. Une erreur `tciError` sera propagée par l'exécutable TE si aucun module de ce genre n'existe.	

7.3.1.1.2 getImportedModules

Signature	`TciModuleIdListType getImportedModules()`
Valeur renvoyée	Liste de tous les modules importés du module radical. Les modules sont ordonnés comme ils apparaissent dans le module TTCN-3. Si aucun module importé n'existe, une liste de modules vide est renvoyée.
Contrainte	Ne doit être utilisé que si un module radical a déjà été sélectionné
Effet	L'exécutable TE fournit à la gestion une liste de modules importés du module radical. Si aucun module importé n'existe, une liste de modules vide est renvoyée. Si l'exécutable TE ne peut pas offrir de liste, la valeur distincte `null` doit être renvoyée.

7.3.1.1.3 tciGetModuleParameters

Signature	`TciModuleParameterListType tciGetModuleParameters(in TciModuleIdType moduleName)`	
Paramètres in	`moduleName`	Le nom `moduleName` indique les identificateurs de module pour lesquels les paramètres de module devraient être extraits.
Valeur renvoyée	Liste de tous les paramètres du module identifié. Ces paramètres sont ordonnés comme ils apparaissent dans le module TTCN-3. Si aucun paramètre n'existe, une liste vide de paramètres de module est renvoyée.	
Contrainte	Ne doit être utilisé que si un module radical a déjà été sélectionné	
Effet	L'exécutable TE fournit à la gestion une liste des paramètres de module du module identifié. Si aucun paramètre de module n'existe, une liste vide de paramètres de module est renvoyée. Si l'exécutable TE ne peut pas offrir de liste, la valeur distincte `null` doit être renvoyée.	

7.3.1.1.4 tciGetTestCases

Signature	`TciTestCaseIdListType tciGetTestCases ()`
Valeur renvoyée	Liste de tous les exécutables de test élémentaire qui sont définis ou importés dans le module radical
Contrainte	Ne doit être utilisé que si un module radical a déjà été sélectionné
Effet	L'exécutable TE fournit à la gestion une liste de tests élémentaires. Si aucun test élémentaire n'existe, une liste vide de tests élémentaires est renvoyée. Si l'exécutable TE ne peut pas offrir de liste, la valeur distincte `null` doit être renvoyée.

7.3.1.1.5 tciGetTestCaseParameters

Signature	`TciParameterTypeListType tciGetTestCaseParameters (in TciTestCaseIdType testCaseId)`	
Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3.
Valeur renvoyée	Liste de tous les types de paramètre du test élémentaire indiqué. Les types de paramètre sont ordonnés comme ils apparaissent dans la signature TTCN-3 du test élémentaire. Si aucun paramètre n'existe, une liste vide de types de paramètre est renvoyée.	
Contrainte	Ne doit être utilisé que si un module radical a déjà été sélectionné	
Effet	L'exécutable TE fournit à la gestion une liste de types de paramètre du test élémentaire indiqué. Si aucun paramètre de test élémentaire n'existe, une liste vide de types de paramètre est renvoyée. Si l'exécutable TE ne peut pas offrir de liste, la valeur distincte `null` doit être renvoyée.	

7.3.1.1.6 tciGetTestCaseTSI

Signature	`TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)`	
Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3.
Valeur renvoyée	Liste de tous les ports du système du test élémentaire indiqué qui ont été déclarés dans la définition du composant systémique pour le test élémentaire, c'est-à-dire de tous les ports d'interface TSI. Si un composant systémique n'a pas été explicitement défini pour le test élémentaire, alors la liste contient tous les ports de communication du composant de test principal (MTC). Les ports sont ordonnés comme ils apparaissent dans la déclaration correspondante de type de composant TTCN-3. Si aucun des ports du système n'existe, une liste vide, c'est-à-dire une liste de longueur zéro, est renvoyée.	
Contrainte	Ne doit être utilisé que si un module radical a déjà été sélectionné	
Effet	L'exécutable TE fournit à la gestion une liste des ports du système du test élémentaire indiqué. Si aucun des ports du système n'existe, une liste de ports vide est renvoyée. Si l'exécutable TE ne peut pas offrir de liste, la valeur distincte `null` doit être renvoyée.	

7.3.1.1.7 tciStartTestCase

Signature	`void tciStartTestCase(in TciTestCaseIdType testCaseId, in TciParameterListType parameterList)`	
Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3
	`parameterList`	Liste de `Values` où chaque valeur définit un paramètre à partir de la liste des paramètres décrite dans la définition du test élémentaire TTCN-3. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la signature TTCN-3 du test élémentaire. Si aucun paramètre ne doit être transmis, la valeur `null` ou une liste vide `parameterList`, c'est-à-dire une liste de longueur zéro, doit être transmise.
Valeur renvoyée	`void`	
Contrainte	Ne doit être appelé que si un module a déjà été sélectionné. Seuls les identificateurs `testCaseId` pour le test élémentaire, qui sont déclarés dans le module TTCN-3 actuellement sélectionné, doivent être transmis. Les tests élémentaires qui sont importés à partir d'un module référencé ne peuvent pas être lancés. Afin de lancer des tests élémentaires importés, le module référencé (importé) doit d'abord être sélectionné au moyen de l'opération tciRootModule.	
Effet	`tciStartTestCase` lance un test élémentaire dans le module actuellement sélectionné avec les paramètres indiqués. Une erreur `tciError` sera propagée par l'exécutable TE si aucun test élémentaire de ce genre n'existe. Tous les paramètres de test élémentaire de type *in* et *inout*, contenus dans la liste `parameterList`, contiennent la valeur `Value`. Tous les paramètres de test élémentaire de type *out*, contenus dans la liste `parameterList, doivent` contenir la valeur distincte `null` étant donné qu'ils ne sont applicables que lorsque le test élémentaire se termine.	

7.3.1.1.8 tciStopTestCase

Signature	`void tciStopTestCase()`
Valeur renvoyée	`void`
Contrainte	Ne doit être appelé que si un module a déjà été sélectionné
Effet	tciStopTestCase arrête le test élémentaire qui est en cours d'exécution. Si l'exécutable TE n'est pas en train d'exécuter un test élémentaire, l'opération sera ignorée. Si la partie commande est en cours d'exécution, l'opération `tciStopTestCase` va arrêter l'exécution du test élémentaire actuellement exécuté, c'est-à-dire arrêtera l'exécution du test élémentaire qui a récemment été indiqué au moyen de l'opération *fournie* `tciTestCaseStarted`. Une partie commande éventuellement en cours d'exécution poursuivra l'exécution comme si le test élémentaire s'était arrêté normalement et avait renvoyé le verdict `ERROR`.

7.3.1.1.9 tciStartControl

Signature	`TriComponentId tciStartControl()`
Valeur renvoyée	Identificateur `TriComponentId` qui représente le composant de test sur lequel la partie commande du module est exécutée. Si l'exécutable TE ne peut pas lancer la partie commande du module sélectionné, la valeur distincte `null` sera renvoyée.
Contrainte	Ne doit être appelé que si un module a déjà été sélectionné
Effet	Lance la partie commande du module sélectionné. La partie commande va lancer le test élémentaire TTCN-3 comme décrit dans la notation TTCN-3. Pendant qu'il est en train d'exécuter la partie commande, l'exécutable TE va appeler les opérations *fournies* `tciTestCaseStarted et tciTestCaseTerminated` pour chaque test élémentaire qui a été lancé et qui s'est terminé. Après la terminaison de la partie commande, l'exécutable TE appelle l'opération *fournie* `TciControlPartTerminated`.

7.3.1.1.10 tciStopControl

Signature	`void tciStopControl()`
Valeur renvoyée	`void`
Contrainte	Ne doit être appelé que si un module a déjà été sélectionné
Effet	`tciStopControl` arrête l'exécution de la partie commande. Si aucune partie commande n'est actuellement en cours d'exécution, l'opération sera ignorée. Si un test élémentaire a été lancé directement, ce test va arrêter l'exécution du test élémentaire en cours comme si l'opération `tciStopTestCase` avait été appelée.

7.3.1.2 Sous-interface TCI-TM fournie

Le présent paragraphe spécifie les opérations que la gestion TM doit fournir à l'exécutable TE.

7.3.1.2.1 tciTestCaseStarted

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

Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3.
	`parameterList`	Liste de valeurs qui font partie de la signature du test élémentaire. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration du test élémentaire TTCN-3.
	`timer`	Valeur en virgule flottante représentant la durée de la temporisation du test élémentaire
Valeur renvoyée	`void`	
Contrainte	Ne doit être appelé qu'après le lancement de la partie commande du module ou d'un test élémentaire au moyen de l'opération *requise* `tciStartControl ou tciStartTestCase`	
Effet	`tciTestCaseStarted` indique à la gestion TM qu'un test élémentaire avec `testCaseId a été` lancé. Aucune différence n'est faite entre le lancement explicite du test élémentaire au moyen de l'opération *requise* `tciStartTestCase` et son lancement implicite pendant qu'il est en train d'exécuter la partie commande.	

7.3.1.2.2 tciTestCaseTerminated

Signature	`void tciTestCaseTerminated(in VerdictValue verdict,` `in TciParameterListType parameterList)`	
Paramètres in	`verdict`	Verdict final du test élémentaire
	`parameterList`	Liste de valeurs qui font partie de la signature du test élémentaire. Les paramètres contenus dans la liste parameterList sont ordonnés comme ils apparaissent dans la déclaration du test élémentaire TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Ne doit être appelé qu'après le lancement de la partie commande du module ou d'un test élémentaire au moyen de l'opération *requise* `tciStartControl ou tciStartTestCase`.	
Effet	Cette opération sera appelée par l'exécutable TE afin d'indiquer à la gestion de test la terminaison du test élémentaire qui a été actuellement exécuté sur le composant MTC et la valeur `verdict` du verdict final. Lors de l'invocation d'une opération `tciTestCaseTerminated`, tous les Paramètres out et *inout* de test élémentaire contiennent des valeurs de type `Value`. Tous les Paramètres in de test élémentaire contiennent la valeur distincte `null` parce qu'ils ne sont applicables qu'au lancement du test élémentaire mais ne le sont pas dans la réponse à cet appel.	

7.3.1.2.3 TciControlTerminated

Signature	`void TciControlTerminated ()`
Valeur renvoyée	`void`
Contrainte	Ne doit être appelé que quand l'exécution du module a été lancée au moyen de l'opération `TciStartControl`
Effet	Cette opération sera appelée par l'exécutable TE afin d'indiquer à la gestion de test que la partie commande du module sélectionné vient juste de finir de s'exécuter.

7.3.1.2.4 tciGetModulePar

Signature	`Value tciGetModulePar (in TciModuleParameterIdType parameterId)`	
Paramètres in	`parameterId`	Identificateur du paramètre de module comme défini dans la notation TTCN-3.
Valeur renvoyée	Une valeur	
Contrainte	Cette opération doit être appelée chaque fois que l'exécutable TE a besoin d'accéder à la valeur d'un paramètre de module. Chaque paramètre de module ayant fait l'objet d'un accès ne sera résolu qu'une seule fois entre une paire `tciStartTestCase` et `tciTestCaseTerminated` si un test élémentaire a été lancé explicitement ou entre une paire `tciStartControl` et `TciControlTerminated` si la partie commande d'un module a été lancée.	
Effet	La gestion fournit à l'exécutable TE une valeur `Value` pour l'identificateur `parameterId` indiqué. Chaque appel d'opération `tciGetModulePar()` va renvoyer la même valeur au cours de toute l'exécution d'un test élémentaire lancé explicitement ou au cours de toute l'exécution d'une partie commande. Si la gestion ne peut pas offrir de valeur TTCN-3, la valeur distincte `null` doit être renvoyée.	

7.3.1.2.5 tciError

Signature	`void tciError(in TString message)`	
Paramètres in	`message`	Valeur de chaîne, c'est-à-dire le message ERROR.
Valeur renvoyée	`void`	
Contrainte	Peut être appelé à tout instant par l'exécutable TE afin d'indiquer une situation d'erreur irrécupérable. Cette situation d'erreur pourrait soit être indiquée par le dispositif CH ou par le codec CD ou pourrait se produire dans l'exécutable TE.	
Effet	L'exécutable TE indique l'apparition d'une situation d'erreur irrécupérable. Le message `message` contient une phrase de cause qui pourrait être communiquée à l'utilisateur du système de test. Il appartient à la gestion de test de mettre fin à l'exécution éventuelle d'un test élémentaire ou d'une partie commande. La gestion de test doit prendre des mesures explicites afin de mettre fin à l'exécution de test immédiatement.	

7.3.2 L'interface TCI-CD

L'interface TCI de codec (TCI-CD) décrit les opérations qu'un exécutable TTCN-3 est tenu d'implémenter et les opérations qu'une implémentation de codec doit fournir à l'exécutable TE pour un certain procédé de codage (Figure 6).

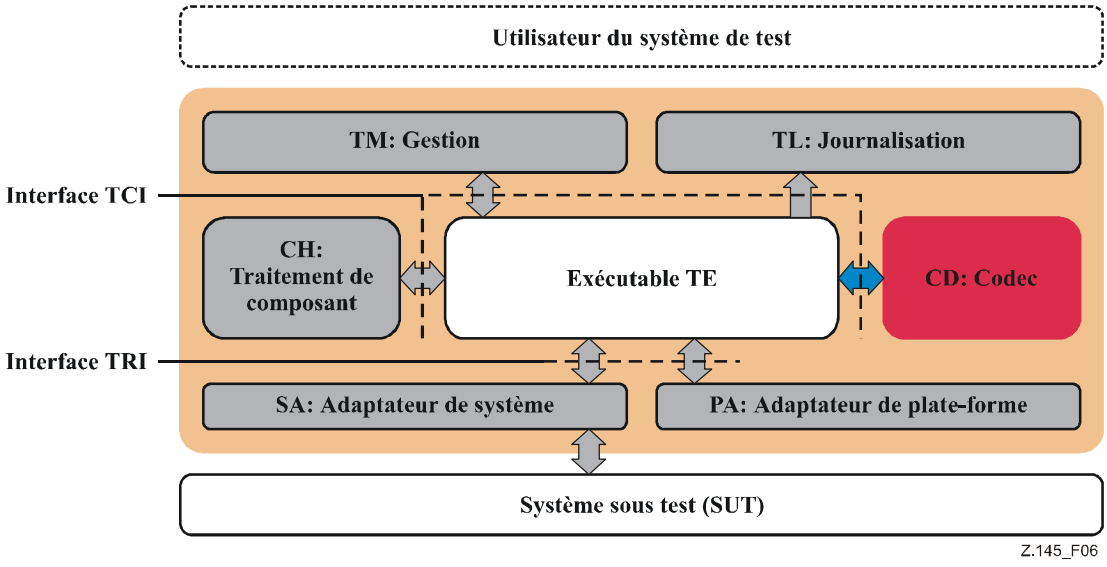

Figure 6/Z.145 – L'interface TCI-CD

Une implémentation de codec code les valeurs TTCN-3 conformément à l'attribut de codage d'une chaîne binaire et décode une chaîne binaire conformément à l'hypothèse de décodage. Pour être en mesure de décoder une chaîne binaire afin d'obtenir une valeur TTCN-3, le codec CD exige une certaine fonctionnalité de la part de l'exécutable TE. Inversement, le codec CD fournit la fonctionnalité de codage et de décodage à l'exécutable TTCN-3.

Le paragraphe 10 illustre l'usage et l'ordonnancement séquentiel des appels d'opération par l'exécutable TE ou par le codec CD.

7.3.2.1 Sous-interface TCI-CD requise

Le présent paragraphe spécifie les opérations que le codec CD exige de l'exécutable TE. Toutes les opérations spécifiées dans le présent paragraphe sont également requises aux sous-interfaces TCI-TM et TCI-CH.

7.3.2.1.1 getTypeForName

Signature	`Type getTypeForName(in TString typeName)`	
Paramètres in	`typeName`	Nom TTCN-3 du type comme défini dans le module TTCN-3. Ci-dessous sont énumérés les noms de type réservés qui renverront un type prédéfini: "integer" "float" "bitstring" "hexstring" "octetstring" "charstring" "universal charstring" "boolean" "verdicttype" "objid" Le nom typeName doit être le nom de type entièrement qualifié, c'est-à-dire le nom module.typeName
Valeur renvoyée	Type représentant le type TTCN-3 demandé	
Contrainte	---	
Effet	Renvoie un type représentant un type TTCN-3. Les types prédéfinis TTCN-3 peuvent être extraits de l'exécutable TE au moyen des mots clés TTCN-3 pour les types prédéfinis. Dans ce cas, le nom `typeName` renvoie au type de base TTCN-3 comme "charstring", "bitstring", etc. Renvoie la valeur distincte `null` si le type demandé ne peut pas être renvoyé. Noter que les types `anytype` et `address` ne peuvent pas être obtenus avec un module réglé à `null`. Bien que ces types soient prédéfinis, ils pourraient être distincts selon le module. Par exemple, le type `address` peut être soit le type prédéfini non modifié, ou un type défini par l'utilisateur dans un module. D'autres types prédéfinis peuvent ne pas être redéfinis.	

7.3.2.1.2 getInteger

Signature	`Type getInteger()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "integer" (entier).
Effet	Construit et renvoie un type de base TTCN-3 "integer" (entier).

7.3.2.1.3 getFloat

Signature	`Type getFloat()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "float" (virgule flottante).
Effet	Construit et renvoie un type de base TTCN-3 "float".

7.3.2.1.4 getBoolean

Signature	`Type getBoolean()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "boolean" (booléen).
Effet	Construit et renvoie un type de base TTCN-3 "boolean".

7.3.2.1.5 getObjid

Signature	`Type getObjid()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "objid" (identificateur d'objet).
Effet	Construit et renvoie un type de base TTCN-3 "objid".

7.3.2.1.6 getCharstring

Signature	`Type getCharstring ()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "charstring" (chaîne de caractères).
Effet	Construit et renvoie un type de base TTCN-3 "charstring".

7.3.2.1.7 getUniversalCharstring

Signature	`Type getUniversalCharstring ()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "universal charstring" (chaîne de caractères universels)
Effet	Construit et renvoie un type de base TTCN-3 "universal charstring".

7.3.2.1.8 getHexstring

Signature	`Type getHexstring ()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "hexstring" (chaîne hexadécimale).
Effet	Construit et renvoie un type de base TTCN-3 "hexstring".

7.3.2.1.9 getBitstring

Signature	`Type getBitstring()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "bitstring" (chaîne binaire).
Effet	Construit et renvoie un type de base TTCN-3 "bitstring".

7.3.2.1.10 getOctetstring

Signature	`Type getOctetstring ()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "octetstring" (chaîne d'octets).
Effet	Construit et renvoie un type de base TTCN-3 "octetstring".

7.3.2.1.11 getVerdict

Signature	`Type getVerdict()`
Valeur renvoyée	Instance de `Type` représentant un type TTCN-3 "verdict".
Effet	Construit et renvoie un type de base TTCN-3 "verdict".

7.3.2.1.12 tciErrorReq

Signature	`void tciErrorReq(in TString message)`	
Paramètres in	`message`	Valeur de chaîne, c'est-à-dire la phrase d'erreur décrivant le problème
Valeur renvoyée	`void`	
Contrainte	Doit être appelé chaque fois qu'une situation d'erreur s'est produite	
Effet	L'exécutable TE sera informé d'une situation d'erreur irrécupérable dans le codec CD et réexpédiera l'indication d'erreur à la gestion de test.	

7.3.2.2 Sous-interface TCI-CD fournie

Le présent paragraphe spécifie les opérations que la gestion TM doit fournir à l'exécutable TE.

7.3.2.2.1 decode

Signature	`Value decode(in le message TriMessageType, in Type decodingHypothesis)`	
Paramètres in	`message`	Le message codé à décoder
	`decodingHypothesis`	L'hypothèse sur laquelle le décodage peut être fondé
Valeur renvoyée	Renvoie la valeur décodée si celle-ci est d'un type compatible en tant qu'hypothèse de décodage `decodingHypothesis`; sinon renvoie la valeur distincte `null`.	
Contrainte	Cette opération doit être appelée chaque fois que l'exécutable TE doit décoder une valeur codée. L'exécutable TE pourra décoder immédiatement après réception d'une valeur codée ou pourra, pour des raisons de performance, différer le décodage jusqu'à ce que la valeur codée fasse l'objet d'un accès réel.	
Effet	Cette opération décode un `message` conformément aux règles de codage et renvoie une valeur TTCN-3. L'hypothèse `decodingHypothesis` doit servir à déterminer si la valeur codée peut être décodée. Si une règle de codage n'est pas autonome c'est-à-dire si le message codé ne contient pas son type par construction, l'hypothèse `decodingHypothesis` doit être utilisée. Si la valeur codée peut être décodée sans l'hypothèse de décodage et si le type déterminé à partir du message codé n'est pas compatible avec l'hypothèse de décodage, la valeur distincte `null` doit être renvoyée.	

7.3.2.2.2 encode

Signature	`TriMessageType encode(in Value value)`	
Paramètres in	`value`	Valeur à coder
Valeur renvoyée	Renvoie un message codé `TriMessage` pour la règle de codage spécifiée	
Contrainte	Cette opération doit être appelée chaque fois que l'exécutable TE doit coder une valeur	
Effet	Renvoie un message codé `TriMessage` conformément aux règles de codage	

7.3.3 L'interface TCI-CH

L'interface TCI de traitement de composant (TCI-CH) décrit les opérations qu'un exécutable TTCN-3 est tenu de mettre en œuvre et les opérations qu'une implémentation de traitement de composant doit fournir à l'exécutable TE (Figure 7).

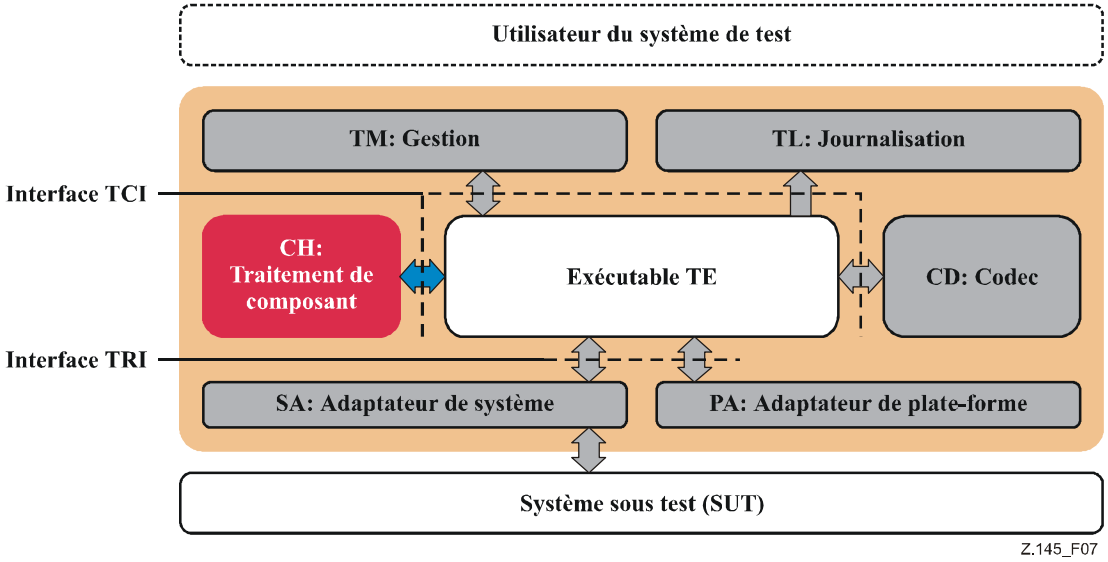

Figure 7/Z.145 – L'interface TCI-CH

Une implémentation de traitement de composant distribue des opérations de configuration TTCN-3 comme `create`, `connect` et `start`, ainsi que la communication entre composants comme l'envoi à un port connecté parmi un ou plusieurs exécutables TTCN-3 participant à une session de test. Noter que, bien que de multiples instances d'un exécutable TE puissent participer à une session de test, cette participation n'est pas obligatoire.

Le principe de base est que la sous-interface TCI-CH n'est pas *en train d'implémenter* une quelconque sorte de fonctionnalité TTCN-3. En revanche, elle sera informée par l'exécutable TE du fait que, p. ex., un composant de test doit être créé. Sur la base de la connaissance interne du traitement de composant (CH, *component handling*), la requête de création d'un composant de test sera transmise à un autre exécutable TE participant (à distance). Ce second exécutable TE participant (à distance) va créer le composant TTCN-3 et va renvoyer un pointeur à l'exécutable TE demandeur (local). L'exécutable TE demandeur (local) peut alors opérer sur le composant de test créé au moyen de ce pointeur de composant.

Dans les définitions d'opération, les termes "exécutable TE local" et "exécutable TE distant" servent à mettre en évidence le fait qu'une implémentation d'un système de test pourrait être répartie entre plusieurs dispositifs de test, chacun d'entre eux hébergeant un exécutable TE local complet. Les termes "local" et "distant" se rapportent toujours aux interfaces qui sont actuellement décrites. Par commodité, le terme "local" se rapporte toujours à l'exécutable TE qui est soit l'appelé d'une opération (pour les *opérations requises*) ou l'appelant d'une opération (pour les *opérations fournies*). Alors que l'exécutable TE est théoriquement considéré comme étant réparti, le dispositif CH est considéré comme étant non réparti. Cette distinction peut être réalisée en utilisant soit une architecture centralisée ou une plate-forme de logiciel médiateur qui donne une vue abstraite à partir des aspects de répartition. Bien que l'exécutable TE puisse être réparti entre différents dispositifs physiques, il pourrait y avoir des configurations où un seul exécutable TE non réparti participera à une session de test. Dans ce cas, les termes "local" et "distant" se rapportent à la même instance d'exécutable TE.

Le paragraphe 10 illustre l'usage et l'ordonnancement séquentiel d'appels d'opération par l'exécutable TE ou par le dispositif CH.

Bien que tous les exécutables TTCN-3 participant à une session de test soient égaux, il y existe un exécutable TE* distinct, qui est celui dans lequel l'opération explicite `tciStartTestCase`() ou `tciStartControl`() a été traitée. La raison de cette distinction est que l'exécutable TE* doit calculer le verdict global, puis informer la gestion de test dès la terminaison de l'exécution de test. Il doit alors fournir le verdict global du test élémentaire.

7.3.3.1 Sous-interface TCI-CH requise

Le présent paragraphe spécifie les opérations que le dispositif CH exige de l'exécutable TE. En plus des opérations spécifiées dans le présent paragraphe, toutes les opérations *requises* de l'interface TCI-CD sont également requises.

7.3.3.1.1 tciEnqueueMsgConnected

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receiver`	Identificateur du composant récepteur
	`rcvdMessage`	Valeur à mettre en file d'attente
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand une opération *fournie* `tciSendConnected` a été appelée dans un exécutable TE distant	
Effet	L'exécutable TE met en file d'attente la valeur reçue dans la file d'attente du port local du composant récepteur indiqué	

7.3.3.1.2 tciEnqueueCallConnected

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receiver`	Identificateur du composant récepteur
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres de valeur qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `TciCallConnected` a été appelée. Tous les paramètres de procédure *in* et *inout* contiennent des valeurs. Tous les paramètres de procédure *out* doivent contenir la valeur distincte `null` parce qu'ils ne sont applicables que dans une réponse à l'appel de procédure mais ne le sont pas dans cet appel de procédure proprement dit. Les paramètres de procédure sont spécifiés dans le modèle de signature TTCN-3.	
Effet	L'exécutable TE met en file d'attente les appels dans la file d'attente du port local du composant récepteur indiqué	

7.3.3.1.3 tciEnqueueReplyConnected

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receiver`	Identificateur du composant recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres de valeur qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
	`returnValue`	Valeur (facultative) de retour de l'appel de procédure
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciReplyConnected` a été appelée. Tous les paramètres de procédure *out* et *inout*, ainsi que la valeur de retour, contiennent des valeurs. Tous les paramètres de procédure *in* doivent contenir la valeur distincte `null`, étant donné qu'ils ne sont applicables qu'à l'appel de procédure mais ne le sont pas dans la réponse à cet appel. La liste `parameterList` contient des paramètres d'appel de procédure. Ces paramètres sont spécifiés dans le modèle de signature TTCN-3. Si aucun type de retour n'a été défini pour la signature de procédure dans la suite ATS en notation TTCN-3, la valeur distincte `null` doit être transmise pour la valeur `returnValue`.	
Effet	L'exécutable TE met en file d'attente la réponse dans la file d'attente du port local du composant récepteur indiqué	

7.3.3.1.4 tciEnqueueRaiseConnected

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receiver`	Identificateur du composant recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`exception`	L'exception
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciRaiseConnected` a été appelée.	
Effet	L'exécutable TE met en file d'attente l'exception dans la file d'attente du port local du composant récepteur indiqué	

7.3.3.1.5 tciCreateTestComponent

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

Paramètres in	`kind`	Sorte du composant qui doit être créé, soit `MTC`, `PTC` ou `CONTROL`.
	`componentType`	Identificateur du type de composant TTCN-3 qui doit être créé
	`name`	Nom du composant qui doit être créé
Valeur renvoyée	Valeur de type `TriComponentIdType` pour le composant créé	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie TciCreateTestComponentReq* a été appelée. L'identificateur `componentType` doit être mis à la valeur distincte `null` si un composant de test de sorte `control` doit être créé. L'identificateur `name` doit être mis à la valeur distincte `null` si aucun nom n'est indiqué dans l'instruction de création TTCN-3.	
Effet	L'exécutable TE crée un composant de test TTCN-3 du type `componentType` et renvoie une référence `TriComponentIdType` au dispositif CH, lequel renvoie la référence à l'exécutable TE distant.	

7.3.3.1.6 tciStartTestComponent

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

Paramètres in	`component`	Identificateur du composant à lancer. Se rapporte à un identificateur déjà créé par un appel de l'opération `TciCreateTestComponent`
	`behaviour`	Identificateur du comportement à lancer sur le composant
	`parameterList`	Liste de valeurs où chaque valeur définit un paramètre à partir de la liste des paramètres décrite dans la déclaration de la fonction TTCN-3 en cours de lancement. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la signature TTCN-3 du test élémentaire. Si aucun paramètre ne doit être transmis, soit la valeur `null` ou une liste vide `parameterList`, c'est-à-dire une liste de longueur zéro, doit être transmise.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciStartTestComponentReq` a été appelée. Etant donné que seuls des Paramètres in sont autorisés pour les fonctions en cours de lancement (Rec. UIT-T Z.140 [2]), la liste `parameterList` contient seulement des Paramètres *in*.	
Effet	L'exécutable TE doit lancer le comportement indiqué dans le composant indiqué	

7.3.3.1.7 tciStopTestComponent

Signature	`void tciStopTestComponent(in TriComponentIdType component)`	
Paramètres in	`component`	Identificateur du composant à arrêter
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciStopTestComponentReq` a été appelée.	
Effet	L'exécutable TE doit arrêter le comportement indiqué dans le composant indiqué	

7.3.3.1.8 tciConnect

Signature	`void tciConnect(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à partir duquel la connexion doit être effectuée
	`toPort`	Identificateur du port de composant de test vers lequel la connexion doit être effectuée
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciConnect` a été appelée.	
Effet	L'exécutable TE doit connecter les ports indiqués les uns aux autres	

7.3.3.1.9 tciDisconnect

Signature	`void tciDisconnect(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à déconnecter
	`toPort`	Identificateur du port de composant de test à déconnecter
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciDisconnect` a été appelée.	
Effet	L'exécutable TE doit déconnecter les ports indiqués	

7.3.3.1.10 tciMap

Signature	`void tciMap(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à partir duquel le mappage doit être effectué
	`toPort`	Identificateur du port de composant de test vers lequel le mappage doit être effectué
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciMapReq` a été appelée.	
Effet	L'exécutable TE doit mapper les ports indiqués les uns vers les autres	

7.3.3.1.11 tciUnmap

Signature	`void tciUnmap(in TriPortIdType fromPort,` `in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à démapper
	`toPort`	Identificateur du port de composant de test à démapper
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciUnmapReq` a été appelée.	
Effet	L'exécutable TE doit démapper les ports indiqués	

7.3.3.1.12 tciTestComponentTerminated

Signature	`void tciTestComponentTerminated(in TriComponentIdType component, in VerdictValue verdict)`	
Paramètres in	`component`	Identificateur du composant qui s'est terminé
	`verdict`	Verdict après la terminaison du composant
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciTestComponentTerminatedReq` a été appelée.	
Effet	L'exécutable TE local est informé de la terminaison du composant de test indiqué dans un exécutable TE distant. Etant donné qu'une fonction qui est exécutée sur un composant de test ne peut avoir que des Paramètres *in* (Rec. UIT-T Z.140 [2], l'opération `TciTestComponentTerminated` n'a pas de paramètre `parameterList`.	

7.3.3.1.13 tciTestComponentRunning

Signature	`TBoolean tciTestComponentRunning (in TriComponentIdType component)`	
Paramètres in	`component`	Identificateur du composant à vérifier quant à son exécution
Valeur renvoyée	`true` si le composant indiqué est encore en train d'exécuter un comportement, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciTestComponentRunningReq` a été appelée.	
Effet	L'exécutable TE local détermine si le composant indiqué est en train d'exécuter un comportement de test. Si le composant est en train d'exécuter un comportement, la valeur `true` sera renvoyée. Dans tout autre cas, p. ex. si le composant de test a terminé son exécution, ou si le composant de test n'a pas été lancé, etc., la valeur `false` sera renvoyée. Après l'envoi du retour d'opération, le dispositif CH va renvoyer la valeur à l'exécutable TE distant.	

7.3.3.1.14 tciTestComponentDone

Signature	`TBoolean tciTestComponentDone (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à sa fin d'exécution
Valeur renvoyée	`true` si le composant indiqué a terminé d'exécuter son comportement, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciTestComponentDoneReq` a été appelée.	
Effet	L'exécutable TE local détermine si le composant indiqué a terminé d'exécuter son comportement de test. Si le composant a terminé son comportement, la valeur `true` sera renvoyée. Dans tout autre cas, p. ex. si le composant de test n'a pas été lancé, ou si le composant de test est encore en cours d'exécution, la valeur `false` sera renvoyée. Après l'envoi du retour d'opération, le dispositif CH va renvoyer la valeur à l'exécutable TE distant.	

7.3.3.1.15 tciGetMTC

Signature	`TriComponentIdType tciGetMTC()`
Valeur renvoyée	Valeur de type `TriComponentIdType` du composant MTC si celui-ci est en train de s'exécuter dans l'exécutable TE local; renvoie la valeur distincte `null` sinon.
Contrainte	Cette opération peut être appelée par le dispositif CH dans l'exécutable TE local approprié quand, dans un exécutable TE distant, une opération *fournie* `tciGetMTCReq` a été appelée.
Effet	L'exécutable TE local détermine si le composant MTC est en train de s'exécuter dans l'exécutable TE local. Si le composant MTC s'exécute dans l'exécutable TE local, l'identificateur du composant MTC sera renvoyé. Si le composant MTC n'est pas exécuté dans l'exécutable TE local, la valeur distincte `null` sera renvoyée. L'opération n'aura aucun effet sur l'exécution du composant MTC. Après l'envoi du retour d'opération, le dispositif CH renverra la valeur à l'exécutable TE distant.

7.3.3.1.16 tciExecuteTestCase

Signature	`void tciExecuteTestCase(in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`	
Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3
	`tsiPortList`	Contient tous les ports qui ont été déclarés dans la définition du composant systémique pour le test élémentaire, c'est-à-dire les ports d'interface TSI. Si `un` composant systémique n'a pas été explicitement défini pour le test élémentaire, alors la liste `tsiPortList` contient tous les ports de communication du composant MTC. Les ports contenus dans la liste `tsiPortList` sont ordonnés comme ils apparaissent dans la déclaration correspondante de type de composant TTCN-3. Si aucun port ne doit être transmis, soit la valeur `null` ou une liste vide `tsiPortList`, c'est-à-dire une liste de longueur zéro, doit être transmise.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local approprié quand, dans un exécutable TE distant, une opération *fournie* `tciExecuteTestCaseReq` a été appelée.	
Effet	L'exécutable TE local détermine s'il y a lieu d'effectuer des connexions statiques au système SUT et l'initialisation de moyens de communication pour des ports d'interface TSI.	

7.3.3.1.17 tciReset

Signature	`void tciReset ()`
Valeur renvoyée	`void`
Contrainte	Cette opération doit être appelée par le dispositif CH dans les exécutables TE locaux appropriés quand, dans un exécutable TE distant, une opération *fournie* `tciResetReq` a été appelée.
Effet	L'exécutable TE peut décider d'utiliser tout moyen permettant de rénitialiser le système de test localement.

7.3.3.1.18 tciKillTestComponent

Signature	`void tciKillTestComponent(in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à supprimer
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciKillTestComponentReq` a été appelée.	
Effet	L'exécutable TE arrête le comportement dans le composant indiqué si nécessaire et le transfère dans l'état supprimé	

7.3.3.1.19 tciTestComponentAlive

Signature	`TBoolean tciTestComponentAlive (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à son existence
Valeur renvoyée	`true` si le composant indiqué est en vie, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciTestComponentAliveReq` a été appelée.	
Effet	L'exécutable TE local détermine si le composant indiqué est en vie. Après l'envoi du retour d'opération, le dispositif CH renverra la valeur à l'exécutable TE distant.	

7.3.3.1.20 tciTestComponentKilled

Signature	`TBoolean tciTestComponentKilled (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à sa non-existence
Valeur renvoyée	`true` si le composant indiqué a été supprimé, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par le dispositif CH dans l'exécutable TE local quand, dans un exécutable TE distant, une opération *fournie* `tciTestComponentKilledReq` a été appelée.	
Effet	L'exécutable TE local détermine si le composant indiqué est dans l'état supprimé. S'il l'est, la valeur `true` sera renvoyée. Dans tout autre cas, la valeur `false` sera renvoyée. Après l'envoi du retour d'opération, le dispositif CH renverra la valeur à l'exécutable TE distant.	

7.3.3.2 TCI-CH fournie

Le présent paragraphe spécifie les opérations que le dispositif CH doit fournir à l'exécutable TE.

7.3.3.2.1 tciSendConnected

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receiver`	Identificateur du composant récepteur
	`sendMessage`	Le message à envoyer
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération d'envoi unidiffusé TTCN-3 à un port de composant qui a été connecté à un autre port de composant.	
Effet	N'envoie une transmission asynchrone qu'au composant récepteur indiqué. Le dispositif CH transmet le message à l'exécutable TE distant dans lequel le récepteur est en cours d'exécution et met en file d'attente les données dans l'exécutable TE distant.	

7.3.3.2.2 tciSendConnectedBC

Signature	`void tciSendConnectedBC(in TriPortIdType sender,` `in Value sendMessage)`	
Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`sendMessage`	Le message à envoyer
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération d'envoi à diffusion sélective TTCN-3 à un port de composant qui a été connecté à d'autres ports de composant.	
Effet	Envoie une transmission asynchrone à tous les composants qui sont connectés à ce port. Le dispositif CH transmet le message à tous les exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente les données dans les exécutables TE distants.	

7.3.3.2.3 tciSendConnectedMC

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receivers`	Identificateurs des composants récepteurs
	`sendMessage`	Le message à envoyer
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 d'envoi en diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant.	
Effet	Envoie une transmission asynchrone à tous les composants récepteurs indiqués. Le dispositif CH transmet le message à tous les exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente les données dans les exécutables TE distants.	

7.3.3.2.4 tciCallConnected

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receiver`	Identificateur du composant récepteur
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres de valeur qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 d'appel unidiffusé à un port de composant qui a été connecté à un autre port de composant. Tous les paramètres de procédure *in* et *inout* contiennent des valeurs. Tous les paramètres de procédure *out* doivent contenir la valeur distincte `null` parce qu'ils ne sont applicables que dans une réponse à l'appel de procédure mais ne le sont pas dans cet appel de procédure proprement dit. Les paramètres de procédure sont spécifiés dans le modèle de signature TTCN-3.	
Effet	Lors de l'invocation de cette opération, l'exécutable TE peut lancer l'appel de procédure correspondant à l'identificateur de signature `signature` dans le composant récepteur appelé. L'opération `tciCallConnected` doit effectuer son retour sans attendre celui de l'appel de procédure propagé. Noter qu'une valeur facultative de fin de temporisation, qui peut être spécifiée dans la suite ATS en notation TTCN-3 pour une opération d'appel, n'est pas incluse dans la signature d'opération `tciCallConnected`. L'exécutable TE est chargé de résoudre ce problème en armant un temporisateur pour l'opération d'appel TTCN-3 dans l'adaptateur PA avec appel d'opération distinct à l'interface TRI, c'est-à-dire `triStartTimer`. Le dispositif CH transmet l'appel à l'exécutable TE distant dans lequel le récepteur est en cours d'exécution et met en file d'attente l'appel dans cet exécutable TE distant.	

7.3.3.2.5 tciCallConnectedBC

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres de valeur qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 d'appel à diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant. Tous les paramètres de procédure *in* et *inout* contiennent des valeurs. Tous les paramètres de procédure *out* doivent contenir la valeur distincte `null` parce qu'ils ne sont applicables que dans une réponse à l'appel de procédure mais ne le sont pas dans cet appel de procédure proprement dit. Les paramètres de procédure sont spécifiés dans le modèle de signature TTCN-3.	
Effet	Lors de l'invocation de cette opération, l'exécutable TE peut lancer l'appel de procédure correspondant à l'identificateur de `signature` signature contenu dans le composant récepteur appelé. L'opération `tciCallConnected` doit effectuer son retour sans attendre celui de l'appel de procédure propagé. Noter qu'une valeur facultative de fin de temporisation, qui peut être spécifiée dans la suite ATS en notation TTCN-3 pour une opération d'appel, n'est pas incluse dans la signature d'opération `tciCallConnected`. L'exécutable TE est chargé de résoudre ce problème en armant un temporisateur pour l'opération d'appel TTCN-3 dans l'adaptateur PA avec appel d'opération distinct à l'interface TRI, c'est-à-dire `triStartTimer`. Le dispositif CH transmet l'appel à tous les exécutables TE distants dans lesquels un récepteur est en cours d'exécution et met en file d'attente l'appel dans ces exécutables TE distants.	

7.3.3.2.6 tciCallConnectedMC

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

Paramètres in	`sender`	Identificateur de port dans le composant émetteur au moyen duquel le message est envoyé
	`receivers`	Identificateur des composants récepteurs
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres de valeur qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 d'appel à diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant. Tous les paramètres de procédure *in* et *inout* contiennent des valeurs. Tous les paramètres de procédure *out* doivent contenir la valeur distincte `null` parce qu'ils ne sont applicables que dans une réponse à l'appel de procédure mais ne le sont pas dans cet appel de procédure proprement dit. Les paramètres de procédure sont spécifiés dans le modèle de signature TTCN-3.	
Effet	Lors de l'invocation de cette opération, l'exécutable TE peut lancer l'appel de procédure correspondant à l'identificateur de signature `signature` contenu dans le composant récepteur appelé. L'opération `tciCallConnected` doit effectuer son retour sans attendre celui de l'appel de procédure propagé. Noter qu'une valeur facultative de fin de temporisation, qui peut être spécifiée dans la suite ATS en notation TTCN-3 pour une opération d'appel, n'est pas incluse dans la signature d'opération `tciCallConnected`. L'exécutable TE est chargé de résoudre ce problème en armant un temporisateur pour l'opération d'appel TTCN-3 dans l'adaptateur PA avec appel d'opération distinct à l'interface TRI, c'est-à-dire `triStartTimer`. Le dispositif CH transmet l'appel à tous les exécutables TE distants dans lesquels un récepteur est en cours d'exécution et met en file d'attente l'appel dans ces exécutables TE distants.	

7.3.3.2.7 tciReplyConnected

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receiver`	Identificateur du composant recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres codés qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
	`returnValue`	Valeur (facultative) de retour de l'appel de procédure.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de réponse unidiffusée à un port de composant qui a été connecté à un autre port de composant. Tous les paramètres de procédure *out* et *inout*, ainsi que la valeur de retour, contiennent des valeurs. Tous les paramètres de procédure *in* doivent contenir la valeur distincte `null`, étant donné qu'ils ne sont applicables qu'à l'appel de procédure mais ne le sont pas dans la réponse à cet appel. La liste `parameterList` contient des paramètres d'appel de procédure. Ces paramètres sont spécifiés dans le modèle de signature TTCN-3. Si aucun type de retour n'a été défini pour la signature de procédure dans la suite ATS en notation TTCN-3, la valeur distincte `null` doit être transmise pour la valeur de retour.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut envoyer la réponse à un appel de procédure correspondant à l'identificateur de signature `signature` et à l'identificateur du composant `receiver`. Le dispositif CH transmet la réponse dans l'exécutable TE distant dans lequel le récepteur est en cours d'exécution et met en file d'attente la réponse dans cet exécutable TE distant.	

7.3.3.2.8 tciReplyConnectedBC

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres codés qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
	`returnValue`	Valeur (facultative) de retour de l'appel de procédure.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de réponse à diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant. Tous les paramètres de procédure *out* et *inout*, ainsi que la valeur de retour, contiennent des valeurs. Tous les paramètres de procédure *in* doivent contenir la valeur distincte `null`, étant donné qu'ils ne sont applicables qu'à l'appel de procédure mais ne le sont pas dans la réponse à cet appel. La liste `parameterList` contient des paramètres d'appel de procédure. Ces paramètres sont spécifiés dans le modèle de signature TTCN-3. Si aucun type de retour n'a été défini pour la signature de procédure dans la suite ATS en notation TTCN-3, la valeur distincte `null` doit être transmise pour la valeur de retour.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut envoyer la réponse à un appel de procédure correspondant à l'identificateur de signature `signature` et tous les composants connectés à l'émetteur `sender`. Le dispositif CH transmet l'exception à tous les exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente l'exception dans ces exécutables TE distants.	

7.3.3.2.9 tciReplyConnectedMC

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receivers`	Identificateur des composants recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`parameterList`	Liste des paramètres codés qui font partie de la signature indiquée. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la déclaration de signature TTCN-3.
	`returnValue`	Valeur (facultative) de retour de l'appel de procédure.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de réponse à diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant. Tous les paramètres de procédure *out* et *inout*, ainsi que la valeur de retour, contiennent des valeurs. Tous les paramètres de procédure *in* doivent contenir la valeur distincte `null`, étant donné qu'ils ne sont applicables qu'à l'appel de procédure mais ne le sont pas dans la réponse à cet appel. La liste `parameterList` contient des paramètres d'appel de procédure. Ces paramètres sont spécifiés dans le modèle de signature TTCN-3. Si aucun type de retour n'a été défini pour la signature de procédure dans la suite ATS en notation TTCN-3, la valeur distincte `null` doit être transmise pour la valeur de retour.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut envoyer la réponse à un appel de procédure correspondant à l'identificateur de signature `signature` et à un des identificateurs de composant contenus dans `receivers`. Le dispositif CH transmet la réponse aux exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente la réponse dans ces exécutables TE distants.	

7.3.3.2.10 tciRaiseConnected

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receiver`	Identificateur du composant recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`exception`	La valeur de l'exception
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de propagation d'exception unidiffusée à un port de composant qui a été connecté à un autre port de composant.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut propager une exception vers un appel de procédure correspondant à l'identificateur de signature `signature` et à l'identificateur du composant `receiver`. Le dispositif CH transmet l'exception à l'exécutable TE distant dans lequel le `receiver` est en cours d'exécution et met en file d'attente l'exception dans cet exécutable TE distant.	

7.3.3.2.11 tciRaiseConnectedBC

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`exception`	La valeur de l'exception
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de propagation d'exception à diffusion sélective à un port de composant qui a été connecté à d'autres ports de composant.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut propager une exception vers un appel de procédure correspondant à l'identificateur de signature `signature` et vers tous les composants connectés à `sender`. Le dispositif CH transmet l'exception à tous les exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente l'exception dans ces exécutables TE distants.	

7.3.3.2.12 tciRaiseConnectedMC

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

Paramètres in	`sender`	Identificateur du port envoyant la réponse
	`receivers`	Identificateurs du composant recevant la réponse
	`signature`	Identificateur de la signature de l'appel de procédure
	`exception`	La valeur de l'exception
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de propagation d'exception à diffusion sélective à un port de composant qui a été connecté à un autre port de composant.	
Effet	Lors de l'invocation de cette opération, le dispositif CH peut propager une exception vers un appel de procédure correspondant à l'identificateur de signature `signature` et à un des identificateurs de composant `receivers`. Le dispositif CH transmet l'exception à tous les exécutables TE distants dans lesquels des récepteurs sont en cours d'exécution et met en file d'attente l'exception dans ces exécutables TE distants.	

7.3.3.2.13 tciCreateTestComponentReq

Signature	`TriComponentIdType tciCreateTestComponentReq(in TciTestComponentKindType kind, in Type componentType, in TString name)`	
Paramètres in	`kind`	Sorte du composant qui doit être créé, soit `MTC`, `PTC` ou `CONTROL`.
	`componentType`	Identificateur du type de composant TTCN-3 qui doit être créé
Valeur renvoyée	Valeur de type `TriComponentIdType` pour le composant créé	
Contrainte	Cette opération doit être appelée à partir de l'exécutable TE quand un composant doit être créé, soit explicitement quand l'opération TTCN-3 de création est appelée ou implicitement quand le composant de test principal (MTC) ou un composant de commande doit être créé. Le nom `name` doit être mis à la valeur distincte `null` si aucun nom n'est indiqué dans l'instruction TTCN-3 de création.	
Effet	Le dispositif CH transmet la requête de création de composant dans l'exécutable TE distant et y appelle l'opération `TciCreateTestComponent` afin d'obtenir un identificateur pour ce composant.	

7.3.3.2.14 tciStartTestComponentReq

Signature	`void tciStartTestComponentReq(in TriComponentIdType component, in TciBehaviourIdType behaviour, in TciParamaterListType parameterList)`	
Paramètres in	`component`	Identificateur du composant à lancer
	`behaviour`	Identificateur du comportement à lancer sur le composant
	`parameterList`	Liste de valeurs où chaque valeur définit un paramètre à partir de la liste des paramètres décrite dans la déclaration TTCN-3 de la fonction en cours de lancement. Les paramètres contenus dans la liste `parameterList` sont ordonnés comme ils apparaissent dans la signature TTCN-3 du test élémentaire. Si aucun paramètre ne doit être transmis, soit la valeur `null` ou une liste vide `parameterList`, c'est-à-dire une liste de longueur zéro, doit être transmise.
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute l'opération TTCN-3 de lancement. Etant donné que seuls des Paramètres in sont autorisés pour les fonctions en cours de lancement (Rec. UIT-T Z.140 [2]), la liste `parameterList` contient seulement des Paramètres in.	
Effet	Le dispositif CH transmet la requête de composant de lancement dans l'exécutable TE distant et y appelle l'opération `tciStartTestComponent`.	

7.3.3.2.15 tciStopTestComponentReq

Signature	`void tciStopTestComponentReq(in TriComponentIdType component)`	
Paramètres in	`component`	Identificateur du composant à arrêter
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute l'opération TTCN-3 d'arrêt.	
Effet	Le dispositif CH transmet la requête de composant d'arrêt dans l'exécutable TE distant et y appelle l'opération `tciStopTestComponent`.	

7.3.3.2.16 tciConnectReq

Signature	`void tciConnectReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à partir duquel la connexion doit être effectuée
	`toPort`	Identificateur du port de composant de test vers lequel la connexion doit être effectuée
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de connexion	
Effet	Le dispositif CH transmet la requête de connexion dans l'exécutable TE distant où il appelle l'opération `tciConnect` afin d'établir une connexion logique entre les deux ports indiqués. Noter que les deux ports peuvent être sur des exécutables TE distants. Dans ce cas, l'envoi du retour d'opération n'a lieu qu'après l'appel de l'opération `tciConnect` sur les deux exécutables TE distants.	

7.3.3.2.17 tciDisconnectReq

Signature	`void tciConnectReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à déconnecter
	`toPort`	Identificateur du port de composant de test à déconnecter
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération de déconnexion TTCN-3	
Effet	Le dispositif CH transmet la requête de déconnexion à l'exécutable TE distant où il appelle l'opération `tciDisconnect` afin de libérer la connexion logique entre les deux ports indiqués. Noter que les deux ports peuvent être sur des exécutables TE distants. Dans ce cas, l'envoi du retour d'opération n'a lieu qu'après l'appel de l'opération `tciDisconnect` sur les deux exécutables TE distants.	

7.3.3.2.18 tciMapReq

Signature	`void tciMapReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à partir duquel le mappage doit être effectué
	`toPort`	Identificateur du port de composant de test vers lequel le mappage doit être effectué
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de mappage.	
Effet	Le dispositif CH transmet la requête de mappage à l'exécutable TE distant où il appelle l'opération `tciMap` afin d'établir un connexion logique entre les deux ports indiqués.	

7.3.3.2.19 tciUnmapReq

Signature	`void tciUnmapReq(in TriPortIdType fromPort, in TriPortIdType toPort)`	
Paramètres in	`fromPort`	Identificateur du port de composant de test à démapper
	`toPort`	Identificateur du port de composant de test à démapper
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de démappage.	
Effet	Le dispositif CH transmet la requête de démappage à l'exécutable TE distant où il appelle l'opération `tciUnmap` afin de libérer la connexion logique entre les deux ports indiqués.	

7.3.3.2.20 tciTestComponentTerminatedReq

Signature	`void tciTestComponentTerminatedReq(in TriComponentIdType component, in VerdictValue verdict)`	
Paramètres in	`composant`	Identificateur du composant qui s'est terminé
	`verdict`	Verdict après la terminaison du composant
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand un composant de test finit de s'exécuter, soit explicitement par l'opération TTNC-3 d'arrêt, ou implicitement s'il a atteint la dernière instruction.	
Effet	Le dispositif CH est informé de la terminaison du composant de test indiqué. Etant donné qu'une fonction qui est exécutée sur un composant de test ne peut avoir que des Paramètres in (Rec. UIT-T Z.140 [2]), l'opération `tciTestComponentTerminateReq` n'a pas de paramètre `parameterList`. Le dispositif CH communique la terminaison du composant indiqué à tous les exécutables TE participants et à l'exécutable TE spécial*, qui garde trace du verdict global.	

7.3.3.2.21 tciTestComponentRunningReq

Signature	`TBoolean tciTestComponentRunningReq (in TriComponentIdType component)`	
Paramètres in	`component`	Identificateur du composant à vérifier quant à son exécution
Valeur renvoyée	`true` si le composant indiqué est encore en train d'exécuter un comportement, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de vérification d'exécution en cours.	
Effet	Le dispositif CH transmet la requête d'exécution en cours à l'exécutable TE distant ayant le composant de test à vérifier, où il appelle l'opération `tciTestComponentRunning` afin de vérifier le statut d'exécution du composant de test indiqué.	

7.3.3.2.22 tciTestComponentDoneReq

Signature	`TBoolean tciTestComponentDoneReq (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à sa fin d'exécution
Valeur renvoyée	`true` si le composant indiqué a terminé d'exécuter son comportement, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de vérification d'exécution terminée `done`.	
Effet	Le dispositif CH transmet la requête de vérification d'exécution terminée à l'exécutable TE distant ayant le composant de test à vérifier, où il appelle l'opération `tciTestComponentDone` afin de vérifier le statut du composant de test indiqué.	

7.3.3.2.23 tciGetMTCReq

Signature	`TriComponentIdType tciGetMTCReq()`
Valeur renvoyée	Valeur de type `TriComponentIdType` du composant MTC.
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute une opération TTCN-3 de composant mtc.
Effet	Le dispositif CH détermine l'identificateur du composant MTC.

7.3.3.2.24 tciExecuteTestCaseReq

Signature	`void tciExecuteTestCaseReq(in TciTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`	
Paramètres in	`testCaseId`	Identificateur de test élémentaire comme défini dans le module TTCN-3
	`tsiPortList`	`tsiPortList` contient tous les ports qui ont été déclarés dans la définition du composant systémique pour le test élémentaire, c'est-à-dire de tous les ports d'interface TSI. Si un composant systémique n'a pas été explicitement défini pour le test élémentaire, alors la liste `tsiPortList` contient tous les ports de communication du composant MTC. Les ports contenus dans la liste `tsiPortList` sont ordonnés comme ils apparaissent dans la déclaration correspondante de type de composant TTCN-3. Si aucun port ne doit être transmis, soit la valeur null ou une liste vide `tsiPortList`, c'est-à-dire une liste de longueur zéro, doit être transmise.
Valeur renvoyée	`void`	
Contrainte	Cette opération peut être appelée par l'exécutable TE immédiatement avant de commencer le comportement de test élémentaire sur le composant MTC (au cours d'une opération TTCN-3 d'exécution).	
Effet	Le dispositif CH transmet la requête d'exécution de test élémentaire aux exécutables TE distants ayant les ports du système du test élémentaire indiqué. Des connexions statiques au système SUT et l'initialisation de moyens de communication pour des ports d'interface TSI peuvent être établies.	

7.3.3.2.25 tciResetReq

Signature	`void tciResetReq()`
Valeur renvoyée	`void`
Contrainte	Cette opération peut être appelée par l'exécutable TE à tout instant afin de réinitialiser le système de test
Effet	Le dispositif CH transmet la requête de réinitialisation à tous les exécutables TE mis en jeu

7.3.3.2.26 tciKillTestComponentReq

Signature	`void tciKillTestComponentReq(in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à supprimer
Valeur renvoyée	`void`	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute l'opération TTCN-3 de suppression `kill`.	
Effet	Le dispositif CH transmet la requête de suppression de composant à l'exécutable TE distant et y appelle l'opération `tciKillTestComponent`.	

7.3.3.2.27 tciTestComponentAliveReq

Signature	`TBoolean tciTestComponentAliveReq (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à son existence
Valeur renvoyée	`true` si le composant indiqué est en vie, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute l'opération TTCN-3 de vérification d'existence `alive`.	
Effet	Le dispositif CH transmet la requête à l'exécutable TE distant qui a créé le composant de test en question, où il appelle l'opération `tciTestComponentAlive` afin de vérifier le statut du composant de test indiqué.	

7.3.3.2.28 tciTestComponentKilledReq

Signature	`TBoolean tciTestComponentKilledReq (in TriComponentIdType comp)`	
Paramètres in	`comp`	Identificateur du composant à vérifier quant à sa non-existence
Valeur renvoyée	`true` si le composant indiqué a été supprimé, la valeur `false` sinon.	
Contrainte	Cette opération doit être appelée par l'exécutable TE quand celui-ci exécute l'opération TTCN-3 de vérification de non-existence `killed`	
Effet	Le dispositif CH transmet la requête à l'exécutable TE distant qui a créé le composant de test en question, où il appelle l'opération `tciTestComponentKilled` afin de vérifier le statut du composant de test indiqué.	

7.3.4 L'interface TCI-TL

L'interface TCI de journalisation de test (TCI-TL) décrit les opérations qu'un exécutable TTCN-3 est tenu d'implémenter et les opérations qu'une implémentation de journalisation de test doit fournir à l'exécutable TE (Figure 8).

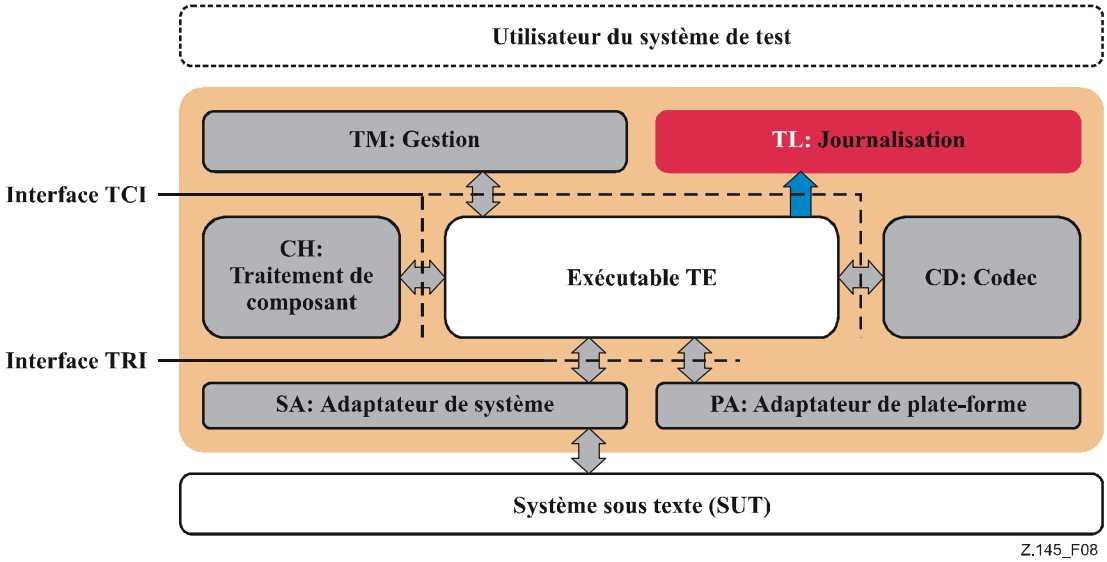

Figure 8/Z.145 – L'interface TCI-TL

La journalisation fournit à l'utilisateur, pour toutes les opérations de niveau TTCN-3, une opération visant à journaliser l'événement qui est effectué, respectivement, par l'exécutable TE, par l'adaptateur SA, par l'adaptateur PA, par le dispositif CH ou par le codec CD.

7.3.4.1 Sous-interface TCI-TL fournie

Le présent paragraphe spécifie les opérations que la journalisation TL doit fournir à l'exécutable TE.

7.3.4.1.1 tliTcExecute

Signature	`void tliTcExecute(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`tcId`	Test élémentaire à exécuter
	`pars`	Liste des paramètres requis par le test élémentaire
	`dur`	Durée de l'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'exécution de la requête de test élémentaire	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.2 tliTcStart

Signature	`void tliTcStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`tcId`	Test élémentaire à exécuter
	`pars`	Liste des paramètres requis par le test élémentaire
	`dur`	Durée de l'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le lancement d'un test élémentaire. Cet événement se produit avant que le test élémentaire soit lancé.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.3 tliTcStop

Signature	`void tliTcStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'arrêt d'un test élémentaire	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.4 tliTcStarted

Signature	`void tliTcStarted(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in TriTimerDurationType dur)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`tcid`	Test élémentaire à exécuter
	`pars`	Liste des paramètres requis par le test élémentaire
	`dur`	Durée de l'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par la gestion TM afin de journaliser le lancement d'un test élémentaire. Cet événement se produit après que le test élémentaire a été lancé.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.5 tliTcTerminated

Signature	`void tliTcTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciTestCaseIdType tcId, in TriParameterListType pars, in VerdictValue outcome)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`tcId`	Test élémentaire à exécuter
	`pars`	Liste des paramètres requis par le test élémentaire
	`outcome`	Verdict de test case
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par la gestion TM afin de journaliser la terminaison d'un test élémentaire. Cet événement se produit après que le test élémentaire s'est terminé.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.6 tliCtrlStart

Signature	`void tliCtrlStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le lancement de la partie commande. Cet événement se produit avant le lancement de la commande. Si la commande n'est pas représentée par un composant d'interface TRI, la valeur c est null.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.7 tliCtrlStop

Signature	`void tliCtrlStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'arrêt de la partie commande. Cet événement se produit avant que la commande soit arrêtée. Si la commande n'est pas représentée par un composant d'interface TRI, la valeur "c" est "null".	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.8 tliCtrlTerminated

Signature	`void tliCtrlTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par la gestion TM afin de journaliser la terminaison de la partie commande. Cet événement se produit après que la commande s'est terminée. Si la commande n'est pas représentée par un composant d'interface TRI, la valeur "c" est "null".	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.9 tliMSend_m

Signature	`void tliMSend_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressType address, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`address`	Adresse de la destination dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`msg`	Message codé
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'envoi unidiffusé. Cet événement se produit après envoi. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.10 tliMSend_m_BC

Signature	`void tliMSend_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`msg`	Message codé
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'envoi à diffusion sélective. Cet événement se produit après envoi. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.11 tliMSend_m_MC

Signature	`void tliMSend_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriMessageType msg, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`addresses`	Adresses des destinations dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`msg`	Message codé
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'envoi en diffusion sélective. Cet événement se produit après envoi. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.12 tliMSend_c

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

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`to`	Composant qui va recevoir le message
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'envoi unidiffusé. Cet événement se produit après envoi. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.13 tliMSend_c_BC

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

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'envoi à diffusion sélective. Cet événement se produit après envoi. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.14 tliMSend_c_MC

Signature	`void tliMSend_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est envoyé
	`msgValue`	Valeur à coder et à envoyer
	`toList`	Composants qui vont recevoir le message
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'envoi en diffusion sélective. Cet événement se produit après envoi. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.15 tliMDetected_m

Signature	`void tliMDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriMessageType msg, in TriAddressType address)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msg`	Message codé qui a été reçu
	`address`	Adresse du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser la mise en file d'attente d'un message. Cet événement se produit après que le message a été mis en file d'attente. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.16 tliMDetected_c

Signature	`void tliMDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TriComponentIdType from)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msgValue`	Message qui a été reçu
	`from`	Composant qui a envoyé le message
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser la mise en file d'attente d'un message. Cet événement se produit après que le message a été mis en file d'attente. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.17 tliMMismatch_m

Signature	`void tliMMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msgValue`	Message qui est comparé au modèle
	`msgTmpl`	Modèle servant à vérifier la concordance du message
	`diffs`	Différence/discordance entre message et modèle
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'un modèle. Cet événement se produit après vérification de la concordance avec un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.18 tliMMismatch_c

Signature	`void tliMMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msgValue`	Message qui est comparé au modèle
	`msgTmpl`	Modèle servant à vérifier la concordance du message
	`diffs`	Différence/discordance entre message et modèle
	`from`	Composant qui a envoyé le message
	`fromTmpl`	Composant expéditeur attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'un modèle. Cet événement se produit après vérification de la concordance avec un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.19 tliMReceive_m

Signature	`void tliMReceive_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msgValue`	Message qui est comparé au modèle
	`msgTmpl`	Modèle servant à vérifier la concordance du message
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la réception d'un message. Cet événement se produit après vérification de la concordance avec un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.20 tliMReceive_c

Signature	`void tliMReceive_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in Value msgValue, in TciValueTemplate msgTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel le message est reçu
	`msgValue`	Message qui est comparé au modèle
	`msgTmpl`	Modèle servant à vérifier la concordance du message
	`from`	Composant qui a envoyé le message
	`fromTmpl`	Composant expéditeur attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la réception d'un message. Cet événement se produit après vérification de la concordance avec un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.21 tliPrCall_m

Signature	`void tliPrCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`address`	Adresse de la destination dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`pars`	Paramètres qui ont été codés
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'appel unidiffusé. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.22 tliPrCall_m_BC

Signature	`void tliPrCall_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`pars`	Paramètres qui ont été codés
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'appel à diffusion sélective. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.23 tliPrCall_m_MC

Signature	`void tliPrCall_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterListType pars, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`addresses`	Adresses des destinations dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`pars`	Paramètres qui ont été codés
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération d'appel à diffusion sélective. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.24 tliPrCall_c

Signature	`void tliPrCall_c(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port,`
	`in TriSignatureIdType signature,`
	`in TciParameterListType parsValue,`
	`in TriComponentIdType to,`
	`in TriStatusType transmissionFailure)`	

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`to`	Composant qui va recevoir le message
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'appel unidiffusé. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.25 tliPrCall_c_BC

Signature	`void tliPrCall_c_BC(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TriPortIdType port,`
	`in TriSignatureIdType signature,`
	`in TciParameterListType parsValue,`
	`in TriStatusType transmissionFailure)`	

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'appel à diffusion sélective. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.26 tliPrCall_c_MC

Signature	`void tliPrCall_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est invoqué
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres de l'opération appelée
	`toList`	Composant qui va recevoir le message
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération d'appel à diffusion sélective. Cet événement se produit après exécution d'un appel. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.27 tliPrGetCallDetected_m

Signature	`void tliPrGetCallDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterListType pars, in TriAddressType address)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'appel détecté
	`pars`	Paramètres qui ont été codés dans l'appel détecté
	`address`	Adresse de la destination dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'opération de mise en file d'attente de traitement d'appel. Cet événement se produit après mise en file d'attente d'un appel. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.28 tliPrGetCallDetected_c

Signature	`void tliPrGetCallDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TriComponentIdType from) in TriComponentIdType from)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'opération appelée
	`parsValue`	Paramètres qui ont été codés dans l'appel détecté
	`from`	Composant qui a appelé l'opération
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'opération de mise en file d'attente de traitement d'appel. Cet événement se produit après mise en file d'attente d'un appel. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.29 tliPrGetCallMismatch_m

Signature	`void tliPrGetCallMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'appel détecté
	`parsValue`	Paramètres contenus dans l'appel détecté
	`parsTmpl`	Modèle servant à vérifier la concordance du paramètre
	`diffs`	Différence/discordance entre appel et modèle
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'un traitement d'appel. Cet événement se produit après vérification du traitement d'appel par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.30 tliPrGetCallMismatch_c

Signature	`void tliPrGetCallMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'appel détecté
	`parsValue`	Paramètres contenus dans l'appel détecté
	`parsTmpl`	Modèle servant à vérifier la concordance du paramètre
	`diffs`	Différence/discordance entre message et modèle
	`from`	Composant qui a appelé l'opération
	`fromTmpl`	Composant appelant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'un traitement d'appel. Cet événement se produit après vérification du traitement d'appel par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.31 tliPrGetCall_m

Signature	`void tliPrGetCall_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'appel détecté
	`parsValue`	Paramètres contenus dans l'appel détecté
	`parsTmpl`	Modèle servant à vérifier la concordance du paramètre
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le traitement d'un appel. Cet événement se produit après concordance du traitement d'appel par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.32 tliPrGetCall_c

Signature	`void tliPrGetCall_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in TciValueTemplate parsTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'appel est reçu
	`signature`	Signature de l'appel détecté
	`parsValue`	Paramètres contenus dans l'appel détecté
	`parsTmpl`	Modèle servant à vérifier la concordance du paramètre
	`from`	Composant qui a appelé l'opération
	`fromTmpl`	Composant appelant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le traitement d'un appel. Cet événement se produit après concordance du traitement d'appel par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.33 tliPrReply_m

Signature	`void tliPrReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriAddressType address, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`address`	Adresse de la destination dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`repl`	Réponse qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de réponse unidiffusée. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.34 tliPrReply_m_BC

Signature	`void tliPrReply_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`repl`	Réponse qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de réponse à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.35 tliPrReply_m_MC

Signature	`void tliPrReply_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriParameterType repl, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`addresses`	Adresses des destinations dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`repl`	Réponse qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de réponse à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.36 tliPrReply_c

Signature	`void tliPrReply_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriComponentIdType to, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`to`	Composant qui va recevoir la réponse
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de réponse unidiffusée. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.37 tliPrReply_c_BC

Signature	`void tliPrReply_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de réponse à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.38 tliPrReply_c_MC

Signature	`void tliPrReply_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value parsValue, in Value replValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est envoyée
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse à envoyer
	`toList`	Composants qui vont recevoir la réponse
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de réponse à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.39 tliPrGetReplyDetected_m

Signature	`void tliPrGetReplyDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriParameterType repl, in TriAddressType address)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`repl`	Réponse codée qui a été reçue
	`address`	Adresse du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'opération de mise en file d'attente de traitement de réponse. Cet événement se produit après mise en file d'attente du traitement de réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.40 tliPrGetReplyDetected_c

Signature	`void tliPrGetReplyDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value replValue, in TriComponentIdType from)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`replValue`	Réponse qui a été reçue
	`from`	Composant qui a envoyé la réponse
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'opération de mise en file d'attente de traitement de réponse. Cet événement se produit après mise en file d'attente du traitement de réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.41 tliPrGetReplyMismatch_m

Signature	`void tliPrGetReplyMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse qui a été reçue
	`replyTmpl`	Modèle servant à vérifier la concordance de la réponse
	`diffs`	Différence/discordance entre réponse et modèl
	`address`	Adresse du fichier source dans le système SU
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'une opération de traitemen t de réponse. Cet événement se produit après vérification du traitement de réponse par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.42 tliPrGetReplyMismatch_c

Signature	`void tliPrGetReplyMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse qui a été reçue
	`replyTmpl`	Modèle servant à vérifier la concordance de la réponse
	`diffs`	Différence/discordance entre réponse et modèle
	`from`	Composant qui a envoyé la réponse
	`fromTmpl`	Composant répondant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'une opération de traitement de réponse. Cet événement se produit après vérification du traitement de réponse par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.43 tliPrGetReply_m

Signature	`void tliPrGetReply_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse qui a été reçue
	`replyTmpl`	Modèle servant à vérifier la concordance de la réponse
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le traitement une réponse. Cet événement se produit après vérification du traitement de réponse par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.44 tliPrGetReply_c

Signature	`void tliPrGetReply_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value replValue, in TciValueTemplate replyTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel la réponse est reçue
	`signature`	Signature se rapportant à la réponse
	`parsValue`	Paramètres de signature se rapportant à la réponse
	`replValue`	Réponse qui a été reçue
	`replyTmpl`	Modèle servant à vérifier la concordance de la réponse
	`from`	Composant qui a envoyé la réponse
	`fromTmpl`	Composant répondant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser le traitement d'une réponse. Cet événement se produit après vérification du traitement de réponse par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.45 tliPrRaise_m

Signature	`void tliPrRaise_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressType address, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`address`	Adresse de la destination dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`exc`	Exception qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de propagation unidiffusée. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.46 tliPrRaise_m_BC

Signature	`void tliPrRaise_m_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`exc`	Exception qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de propagation à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.47 tliPrRaise_m_MC

Signature	`void tliPrRaise_m_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriAddressListType addresses, in TriStatusType encoderFailure, in TriExceptionType exc, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`addresses`	Adresses des destinations dans le système SUT
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`exc`	Exception qui a été codée
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser une opération de propagation à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.48 tliPrRaise_c

Signature	`void tliPrRaise_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdType to, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`to`	Composant qui va recevoir la réponse
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de propagation unidiffusée. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.49 tliPrRaise_c_BC

Signature	`void tliPrRaise_c_BC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de propagation à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.50 tliPrRaise_c_MC

Signature	`void tliPrRaise_c_MC(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TriComponentIdListType toList, in TriStatusType transmissionFailure)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est envoyée
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception à envoyer
	`toList`	Composants qui vont recevoir la réponse
	`transmissionFailure`	Message de défaillance qui pourrait apparaître lors de la transmission
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser une opération de propagation à diffusion sélective. Cet événement se produit après exécution de la réponse. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.51 tliPrCatchDetected_m

Signature	`void tliPrCatchDetected_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TriExceptionType exc, in TriAddressType address)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`exc`	Exception qui a été acquise
	`address`	Adresse du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'opération de mise en file d'attente d'acquisition. Cet événement se produit après mise en file d'attente des acquisitions. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.52 tliPrCatchDetected_c

Signature	`void tliPrCatchDetected_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in Value excValue, in TriAddressType address)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`excValue`	Exception qui a été acquise
	`address`	Adresse du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'opération de mise en file d'attente d'acquisition. Cet événement se produit après mise en file d'attente des acquisitions. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.53 tliPrCatchMismatch_m

Signature	`void tliPrCatchMismatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception qui a été reçue
	`excTmpl`	Modèle servant à vérifier la concordance de l'exception
	`diffs`	Différence/discordance entre exception et modèle
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'une opération d'acquisition. Cet événement se produit après vérification d'une acquisition par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.54 tliPrCatchMismatch_c

Signature	`void tliPrCatchMismatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TciValueDifferenceList diffs, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception qui a été reçue
	`excTmpl`	Modèle servant à vérifier la concordance de l'exception
	`diffs`	Différence/discordance entre exception et modèle
	`from`	Composant qui a envoyé la réponse
	`fromTmpl`	Composant répondant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'un opération d'acquisition. Cet événement se produit après vérification de l'acquisition par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.55 tliPrCatch_m

Signature	`void tliPrCatch_m(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriAddressType address, in TciValueTemplate addressTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception qui a été reçue
	`excTmpl`	Modèle servant à vérifier la concordance de l'exception
	`address`	Adresse du fichier source dans le système SUT
	`addressTmpl`	Adresse attendue du fichier source dans le système SUT
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'acquisition d'une exception. Cet événement se produit après vérification de l'acquisition par rapport à un modèle. Cet événement sert à journaliser la communication avec le système SUT.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.56 tliPrCatch_c

Signature	`void tliPrCatch_c(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature, in TciParameterListType parsValue, in Value excValue, in TciValueTemplate excTmpl, in TriComponentIdType from, in TciNonValueTemplate fromTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
	`parsValue`	Paramètres de signature se rapportant à l'exception
	`excValue`	Exception qui a été reçue
	`excTmpl`	Modèle servant à vérifier la concordance de l'exception
	`from`	Composant qui a envoyé la réponse
	`fromTmpl`	Composant répondant qui est attendu
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'acquisition d'une exception. Cet événement se produit après vérification de l'acquisition par rapport à un modèle. Cet événement sert à journaliser la communication entre composants.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.57 tliPrCatchTimeoutDetected

Signature	`void tliPrCatchTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser la détection d'une fin de temporisation d'acquisition. Cet événement se produit après mise en file d'attente de la fin de temporisation.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.58 tliPrCatchTimeout

Signature	`void tliPrCatchTimeout(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port, in TriSignatureIdType signature)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port au moyen duquel l'exception est reçue
	`signature`	Signature se rapportant à l'exception
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'acquisition d'une temporisation. Cet événement se produit après que l'acquisition de temporisation a été effectuée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.59 tliCCreate

Signature	`void tliCCreate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TString name)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est créé
	`nom`	Nom du composant qui est créé
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de création de composant. Cet événement se produit après création de composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.60 tliCStart

Signature	`void tliCStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciBehaviourIdType beh, in TciParameterListType pars)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est lancé
	`beh`	Comportement en cours de lancement sur le composant
	`pars`	Paramètres du comportement lancé
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de lancement de composant. Cet événement se produit après lancement d'un composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.61 tliCRunning

Signature	`void tliCRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est vérifié quant à son exécution en cours
	`status`	Statut de ce composant
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de composant en cours d'exécution. Cet événement se produit après le lancement de l'exécution d'un composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.62 tliCAlive

Signature	`void tliCAlive(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est vérifié quant à son exécution
	`status`	Statut de ce composant
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de vérification de l'existence d'un composant. Cet événement se produit après l'existence d'un composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.63 tliCStop

Signature	`void tliCStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est arrêté
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération d'arrêt d'un composant. Cet événement se produit après l'arrêt d'un composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.64 tliCKill

Signature	`void tliCKill(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`comp`	Composant qui est arrêté
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de suppression d'un composant. Cet événement se produit après suppression d'un composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.65 tliCDoneMismatch

Signature	`void tliCDoneMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`compTmpl`	Modèle servant à vérifier la concordance avec une fin d'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'une opération de vérification de fin d'exécution d'un composant. Cet événement se produit après vérification de la fin d'exécution par rapport à un modèle.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.66 tliCDone

Signature	`void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType comp, in TciNonValueTemplate compTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`compTmpl`	Modèle servant à vérifier la concordance avec une fin d'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de vérification de fin d'exécution d'un composant. Cet événement se produit après l'opération de vérification de fin d'exécution.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.67 tliCKilledMismatch

Signature	`void tliCKilledMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`compTmpl`	Modèle servant à vérifier la concordance avec une fin d'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la discordance d'une opération de composant supprimé. Cet événement se produit après que l'état supprimé est vérifié par rapport à un modèle.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.68 tliCKilled

Signature	`void tliCKilled(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate compTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`compTmpl`	Modèle servant à vérifier la concordance avec une fin d'exécution
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de composant supprimé. Cet événement se produit après l'opération de composant supprimé.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.69 tliCTerminated

Signature	`void tliCTerminated(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`verdict`	Verdict du composant
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la terminaison d'un composant. Cet événement se produit après la terminaison du composant.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.70 tliPConnect

Signature	`void tliPConnect(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`c1`	Composant du premier port à connecter
	`port1`	Premier port à connecter
	`c2`	Composant du second port à connecter
	`port2`	Second port à connecter
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'opération de connexion. Cet événement se produit après l'opération de connexion.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.71 tliPDisconnect

Signature	`void tliPDisconnect(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`c1`	Composant du premier port à déconnecter
	`port1`	Premier port à déconnecter
	`c2`	Composant du second port à déconnecter
	`port2`	Second port à déconnecter
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le dispositif CH afin de journaliser l'opération de déconnexion. Cet événement se produit après l'opération de déconnexion.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.72 tliPMap

Signature	`void tliPMap(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`c1`	Composant du premier port à mapper
	`port1`	Premier port à mapper
	`c2`	Composant du second port à mapper
	`port2`	Second port à mapper
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'opération de mappage. Cet événement se produit après l'opération de mappage.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.73 tliPUnmap

Signature	`void tliPUnmap(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1, in TriComponentIdType c2, in TriPortIdType port2)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`c1`	Composant du premier port à démapper
	`port1`	Premier port à démapper
	`c2`	Composant du second port à démapper
	`port2`	Second port à démapper
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur SA afin de journaliser l'opération de démappage. Cet événement se produit après l'opération de démappage.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.74 tliPClear

Signature	`void tliPClear(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port à libérer
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de libération de port. Cet événement se produit après l'opération de libération de port.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.75 tliPStart

Signature	`void tliPStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port à lancer
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de lancement de port. Cet événement se produit après l'opération de lancement de port.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.76 tliPStop

Signature	`void tliPStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port à arrêter.
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération d'arrêt de port. Cet événement se produit après l'opération d'arrêt de port.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.77 tliPHalt

Signature	`void tliPHalt(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriPortIdType port)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`port`	Port à arrêter
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de mise en pause d'un port. Cet événement se produit après l'opération de mise en pause d'un port.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.78 tliEncode

Signature	`void tliEncode(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType encoderFailure, in TriMessageType msg, in TString codec)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`value`	Valeur à coder
	`encoderFailure`	Message de défaillance qui pourrait apparaître lors du codage
	`msg`	Valeur codée
	`codec`	Codeur utilisé
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le codec CD afin de journaliser l'opération de codage.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.79 tliDecode

Signature	`void tliDecode(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value val, in TriStatusType decoderFailure, in TriMessageType msg, in TString codec)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`msg`	Valeur à décoder
	`decoderFailure`	Message de défaillance qui pourraient se produire à décodage
	`value`	Valeur décodée
	`codec`	Décodeur utilisé
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par le codec CD afin de journaliser l'opération de décodage.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.80 tliTTimeoutDetected

Signature	`void tliTTimeoutDetected(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timer`	Temporisateur qui est arrivé à expiration
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser la détection d'une fin de temporisation. Cet événement se produit après la mise en file d'attente d'une fin de temporisation.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.81 tliTTimeoutMismatch

Signature	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timerTmpl`	Modèle de temporisateur qui n'a pas été en concordance
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser une discordance de temporisation. Cet événement se produit après un échec de concordance de fin de temporisation.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.82 tliTTimeoutMismatch

Signature	`void tliTTimeoutMismatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciNonValueTemplate timerTmpl)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timerTmpl`	Modèle de temporisateur qui a été en concordance
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser une concordance de temporisation. Cet événement se produit après qu'une temporisation a été en concordance.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.83 tliTStart

Signature	`void tliTStart(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType dur)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timer`	Temporisateur qui est armé
	`dur`	Durée de la temporisation
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser l'armement d'un temporisateur. Cet événement se produit après l'opération d'armement de temporisateur.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.84 tliTStop

Signature	`void tliTStop(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timer`	Temporisateur qui est arrêté
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser l'arrêt d'un temporisateur. Cet événement se produit après l'opération d'arrêt d'un temporisateur.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.85 tliTRead

Signature	`void tliTRead(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TriTimerDurationType elapsed)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timer`	Temporisateur qui est armé
	`elapsed`	Temps écoulé dans la temporisation
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser la lecture d'un temporisateur. Cet événement se produit après l'opération de lecture de temporisateur.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.86 tliTRunning

Signature	`void tliTRunning(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`timer`	Temporisateur qui est vérifié quant à son exécution
	`status`	Statut de ce composant
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'adaptateur PA afin de journaliser l'opération d'exécution de temporisation. Cet événement se produit après l'opération d'exécution de temporisation.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.87 tliSEnter

Signature	`void tliSEnter(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TString name, in TciParameterListType parsValue,`
	`in TString kind)`	

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`name`	Nom de la portée
	`parsValue`	Paramètres de la portée
	`kind`	Sorte de la portée
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'entrée dans une portée. Cet événement se produit après l'entrée dans la portée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.88 tliSLeave

Signature	`void tliSLeave(in TString am, in TInteger ts, in TString src,`
	`in TInteger line, in TriComponentIdType c,`
	`in TString name, in Value val, in TString kind)`	

Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`name`	Nom de la portée
	`val`	Valeur de retour de la portée
	`kind`	Sorte de la portée
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la sortie d'une portée. Cet événement se produit après la sortie de la portée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.89 tliVar

Signature	`void tliVar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value varValue)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`name`	Nom de la variable
	`varValue`	Nouvelle valeur de la variable
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la modification de la valeur d'une variable. Cet événement se produit après modification des valeurs.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.90 tliModulePar

Signature	`void tliModulePar(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in Value parValue)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`name`	Nom du paramètre de module
	`parValue`	Valeur du paramètre de module
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la valeur d'un paramètre de module. Cet événement se produit après accès à la valeur d'un paramètre de module.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.91 tliGetVerdict

Signature	`void tliGetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`verdict`	Valeur actuelle du verdict local
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de requête de verdict. Cet événement se produit après l'opération de requête de verdict.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.92 tliSetVerdict

Signature	`void tliSetVerdict(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in VerdictValue verdict)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`verdict`	Valeur à attribuer au verdict local
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'opération de mise à jour d'un verdict. Cet événement se produit après l'opération de mise à jour d'un verdict.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.93 tliLog

Signature	`void tliLog(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TciValueList log)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`log`	Chaîne à journaliser
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par la gestion TM afin de journaliser l'opération TTCN-3 de journalisation d'instruction. Cet événement se produit après l'opération TTCN-3 de journalisation.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.94 tliAEnter

Signature	`void tliAEnter(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'entrée dans une instruction "alt". Cet événement se produit après qu'une instruction "alt" a fait l'objet d'une entrée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.95 tliALeave

Signature	`void tliALeave(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la sortie d'une instruction "alt". Cet événement se produit après la sortie de l'instruction "alt".	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.96 tliANomatch

Signature	`void tliANomatch(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la non-concordance d'une instruction "alt". Cet événement se produit après que l'instruction "alt" n'a pas été en concordance.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.97 tliARepeat

Signature	`void tliARepeat(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la répétition d'une instruction "alt". Cet événement se produit quand l'instruction "alt" a été répétée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.98 tliADefaults

Signature	`void tliADefaults(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'entrée dans la section par défaut. Cet événement se produit après que la section par défaut a fait l'objet d'une entrée.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.99 tliAActivate

Signature	`void tliAActivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in TString name, in TciParameterListType pars, in Value ref)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`name`	Nom de la valeur par défaut
	`pars`	Paramètre de la valeur par défaut
	`ref`	Référence de la valeur par défaut résultante
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser l'activation d'une valeur par défaut. Cet événement se produit après l'activation d'une valeur par défaut.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.100 tliADeactivate

Signature	`void tliADeactivate(in TString am, in TInteger ts, in TString src, in TInteger line, in TriComponentIdType c, in Value ref)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
	`ref`	Référence de la valeur par défaut résultante
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser la désactivation d'une valeur par défaut. Cet événement se produit après la désactivation de valeur par défaut.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

7.3.4.1.101 tliAWait

Signature	`void tliAWait(in string am, in long ts, in string src, in long line, in TriComponentId c)`	
Paramètres in	`am`	Message additionnel
	`ts`	Instant auquel l'événement est produit
	`src`	Fichier source de la spécification de test
	`line`	Numéro de la ligne où la requête est effectuée
	`c`	Composant qui produit cet événement
Valeur renvoyée	`void`	
Contrainte	Doit être appelé par l'exécutable TE afin de journaliser que le composant attend des événements pour un nouvel instantané.	
Effet	L'interface de journalisation TL présente à l'utilisateur toutes les informations fournies dans les paramètres de cette opération. La méthode correspondante est hors du domaine d'application de la présente Recommandation.	

8 Mappage vers le langage Java

8.1 Introduction

Le présent paragraphe introduit le mappage vers le langage Java à l'interface TCI. Pour des raisons d'efficacité, un mappage linguistique spécial est en revanche introduit, allant du langage IDL (du groupe OMG) vers le langage Java.

Le mappage vers le langage Java à l'interface de commande TTCN-3 précise comment les définitions du langage IDL, décrites dans le paragraphe 7, sont mappées vers le langage Java. Le mappage linguistique est indépendant de la version Java utilisée car seules les constructions syntaxiques de base du langage Java sont utilisées.

8.2 Noms et domaines d'application

8.2.1 Noms

Bien qu'il n'y ait aucun conflit entre les identificateurs utilisés dans la définition IDL et le langage Java, certaines règles de conversion nominative sont appliquées aux identificateurs IDL.

Les interfaces ou identificateurs de classe Java omettent le type final `Type` utilisé dans la définition IDL.

EXEMPLE: le type IDL **`TciTestCaseIdType`** correspond à `TciTestCaseId` dans le langage Java.

Le mappage résultant est conforme aux conventions de codage normales du langage Java.

8.2.2 Portées

Le module IDL **`tciInterface`** est mappé vers le paquetage Java `org.etsi.ttcn3.tci`. Toutes les déclarations de type IDL contenues dans ce module sont mappées vers les classes ou déclarations d'interface Java contenues dans ce paquetage.

Mappage de type

Mappage de type de base

Le Tableau 2 donne un aperçu général de la façon dont les types originels TBoolean, TFloat, TInteger, TString et TStringseq sont mappés vers les types Java.

Tableau 2/Z.145 – Mappages de type de base

Type IDL	Type Java
TBoolean	boolean
TFloat	float
TInteger	int
TString	java.lang.String
TStringseq	java.lang.String[]

Le type originel TObjId est défini dans le paragraphe correspondant de l'interface ObjidValue.

boolean

Le type IDL **TBoolean** est mappé vers le type Java de base boolean.

float

Le type IDL **TFloat** est mappé vers le type Java de base float.

char

Le type IDL **TChar** est mappé vers le type Java de base char.

int

Le type IDL **TInteger** est mappé vers le type Java de base int.

String

Le type IDL **TString** est mappé vers la classe java.lang.String sans vérification ni bornes d'étendue des caractères contenus dans la chaîne. Toutes les chaînes pouvant être définies dans la notation TTCN-3 peuvent être converties en classe java.lang.String.

String[]

Le type IDL **TStringseq** est mappé vers une liste tabulaire de la classe java.lang.String.

Universal Char

Le type IDL **TUniversalChar** est mappé vers un type java de base int. L'entier utilise la forme canonique comme défini au § 6.2/X.290.

Mappage de type structuré

La description du langage IDL à l'interface TCI détermine les types définis par l'utilisateur comme étant des types originels. Dans le mappage vers le langage Java, ces types sont mappés vers les interfaces Java. Ces interfaces définissent les méthodes et les attributs qui sont disponibles pour les objets implémentant cette interface.

8.2.2.1 TciParameterType

TciParameterType est mappé vers l'interface suivante:

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

Méthodes:

- `getParameterName()` — Renvoie le nom du paramètre comme défini dans la spécification TTCN-3;
- `setParameterName(String name)` — Met le nom de ce paramètre `TciParameter` à la valeur `name`;
- `getParameterPassingMode()` — Renvoie le mode de communication de ce paramètre;
- `setParameterPassingMode(TciParameterPassingMode mode)` — Met le mode paramétrique de ce paramètre `TriParameter` à la valeur `mode`;
- `getParameter()` — Renvoie le paramètre de valeur de ce paramètre TciParameter, ou l'objet `null` si le paramètre contient la valeur distincte `null`;
- `setParameter(Value parameter)` — Met le paramètre de valeur de ce paramètre TciParameter à parameter. Si la valeur distincte `null` doit être mise afin d'indiquer que ce paramètre ne contient aucune valeur, la valeur Java `null` doit être transmise comme paramètre.

8.2.2.2 TciParameterPassingModeType

`TciParameterPassingModeType` est mappé vers l'interface suivante:

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

Constantes:

- `TCI_IN` — Servira à indiquer qu'un paramètre `TciParameter` est un paramètre `in`;
- `TCI_INOUT` — Servira à indiquer qu'un paramètre `TciParameter` est un paramètre `inout`;
- `TCI_OUT` — Servira à indiquer qu'un paramètre `TciParameter` est un paramètre `out`.

8.2.2.3 TciParameterListType

`TciParameterListType` est mappé vers l'interface suivante:

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

Méthodes:

- `size()` — Renvoie le nombre de paramètres contenus dans cette liste;
- `isEmpty()` — Renvoie la valeur `true` si cette liste ne contient aucun paramètre;
- `getParameters()` — Renvoie une valeur `Enumeration` concernant les paramètres contenus dans la liste. L'énumération fournit les paramètres dans l'ordre où ils apparaissent dans la liste;
- `get(int index)` — Renvoie le paramètre `TciParameter` à la position spécifiée;
- `clear()` — Retire tous les paramètres de cette liste `TciParameterList`;
- `add(TciParameter parameter)` — Ajoute la valeur par`ameter` à la fin de cette liste `TciParameterList`;

- `setParameter(TciParameter[] parameters)`
  Remplit cette liste `TciParameterList` avec les valeurs `parameter`.

8.2.2.4 TciTypeClassType

`TciTypeClassType` est mappé vers l'interface suivante:

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

8.2.2.5 TciTestComponentKindType

`TciTestComponentKindType` est mappé vers l'interface suivante:

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

8.2.2.6 TciBehaviourIdType

`TciBehaviourIdType` est mappé vers l'interface suivante:

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

8.2.2.7 TciTestCaseIdType

`TciTestCaseIdType` est mappé vers l'interface suivante:

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

8.2.2.8 TciModuleIdType

`TciModuleIdType` est mappé vers l'interface suivante:

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

8.2.2.9 TciModuleParameterIdType

`TciModuleParameterIdType` est mappé vers l'interface suivante:

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

8.2.2.10 TciModuleParameterListType

`TciModuleParameterListType` est mappé vers l'interface suivante:

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

Méthodes:

- `size()` Renvoie le nombre de paramètres de module contenus dans cette liste;
- `isEmpty()` Renvoie la valeur `true` si cette liste ne contient aucun paramètre de module;
- `getModuleParameters()` Renvoie un type `Enumeration` concernant les paramètres de module dans la liste. L'énumération fournit les paramètres de module dans l'ordre de leur apparition dans la liste;
- `get(int index)` Renvoie le paramètre `TciModuleParameter` à la position spécifiée.

8.2.2.11 TciModuleParameterType

`TciModuleParameterType` est mappé vers l'interface suivante:

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

Méthodes:

- `getModuleParameterName ()` Renvoie le nom du paramètre de module comme défini dans la spécification TTCN-3;
- `getDefaultValue()` Renvoie le paramètre de module par défaut `Value` de ce paramètre `TciModuleParameter`, ou l'objet `null` si le paramètre de module contient la valeur distincte `null`.

8.2.2.12 TciParameterTypeListType

`TciParameterTypeListType` est mappé vers l'interface suivante:

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

Méthodes:

- `size()` Renvoie le nombre de types de paramètre contenus dans cette liste;
- `isEmpty()` Renvoie la valeur `true` si cette liste ne contient aucun type de paramètre;

- `getParameterTypes()` Renvoie un type `Enumeration` concernant les types de paramètre contenus dans la liste. L'énumération fournit les types de paramètre dans l'ordre de leur apparition dans la liste;
- `get(int index)` Renvoie le paramètre `TciParameterType` à la position spécifiée.

8.2.2.13 TciModuleIdListType

`TciModuleIdListType` est mappé vers l'interface suivante:

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

Méthodes:

- `size()` Renvoie le nombre de modules contenus dans cette liste;
- `isEmpty()` Renvoie la valeur `true` si cette liste ne contient aucun module;
- `getImportedModules()` Renvoie une `Enumeration` concernant les modules contenus dans la liste. L'énumération fournit les modules dans l'ordre de leur apparition dans la liste
- liste`get(int index)` Renvoie le paramètre `TciModuleId` à la position spécifiée.

8.2.3 Mappage de type abstrait

Les types TTCN-3 de données sont modélisés dans le langage Java au moyen du mappage de type abstrait comme défini dans le présent article. L'interface `Type` définit seulement les opérations servant à extraire des valeurs des types définis dans la notation TTCN-3 . Aucun type TTCN-3 ne peut être construit au moyen de l'interface `Type`. Les types sont modélisés au moyen de l'interface unique `Type`, qui fournit des méthodes permettant d'identifier des types et d'extraire des valeurs d'un type indiqué.

8.2.3.1 Type

`Type` est mappé vers l'interface suivante:

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

Méthodes:

- `getDefiningModule()` Renvoie l'identificateur du module dans lequel le type a été défini. Si le type représente un type de base TTCN-3, la valeur distincte `null` sera renvoyée;
- `getName()` Renvoie le nom du type comme défini dans le module TTCN-3;
- `getTypeClass()` Renvoie la classe du type considéré. Une valeur `TciTypeClassType` peut avoir une des constantes suivantes:
  `ADDRESS, ANYTYPE, BITSTRING, BOOLEAN, CHARSTRING, COMPONENT, ENUMERATED, FLOAT, HEXSTRING, INTEGER, OBJID, OCTETSTRING, RECORD, RECORD_OF, SET, SET_OF, UNION, UNIVERSAL_CHARSTRING, VERDICT`;
- `newInstance()` Renvoie une valeur récemment créée du type indiqué. Cette valeur initiale de la valeur créée est indéfinie;
- `getTypeEncoding()` Renvoie l'attribut de codage du type comme défini dans le module TTCN-3;

- `getTypeEncodingVariant()` Cette opération renvoie l'attribut de variante de codage de valeur comme défini dans la notation TTCN-3, s'il existe. Si aucun attribut de variante de codage n'a été défini, la valeur distincte `null` sera renvoyée;
- `getTypeEncoding()` Renvoie l'attribut d'extension de type comme défini dans le module TTCN-3.

8.2.4 Mappage de valeur abstraite

Des valeurs TTCN-3 peuvent être extraites de l'exécutable TE et construites au moyen de l'interface Value. L'interface de mappage de valeur est construite hiérarchiquement avec Value comme interface de base. Des interfaces spécialisées ont été définies pour différents types de valeur.

8.2.4.1 Value

`Value`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getType()` Renvoie le type de la valeur spécifiée;
- `notPresent()` Renvoie la valeur `true` si la valeur spécifiée est `omit`, la valeur `false` sinon;
- `getValueEncoding()` Cette opération renvoie l'attribut de codage de valeur comme défini dans la notation TTCN-3, s'il existe. Si aucun attribut de codage n'a été défini la valeur distincte `null` sera renvoyée;
- `getValueEncodingVariant()` Cette opération renvoie l'attribut de variante de codage de valeur comme défini dans la notation TTCN-3, s'il existe. Si aucun attribut de variante de codage n'a été défini, la valeur distincte value `null` sera renvoyée.

8.2.4.2 IntegerValue

`IntegerValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `setInteger(int value)` Met cette valeur IntegerValue à la valeur d'entier `value`;
- `getInteger()` Renvoie la valeur d'entier représentée par ce paramètre IntegerValue.

8.2.4.3 FloatValue

`FloatValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `setFloat(float value)` — Met ce paramètre FloatValue à la valeur en virgule flottante `value`;
- `getFloat()` — Renvoie la valeur en virgule flottante représentée par ce paramètre FloatValue.

8.2.4.4 BooleanValue

`BooleanValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `setBoolean(boolean value)` — Met ce type BooleanValue à la valeur booléenne `value`;
- `getBoolean()` — Renvoie la valeur booléenne représentée par ce type BooleanValue.

8.2.4.5 ObjidValue

`ObjidValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getObjid()` — Renvoie la valeur d'identification d'objet du type TTCN-3 `objid`;
- `setObjid(TciObjId value)` — Met ce paramètre `ObjidValue` à la valeur `value`.

8.2.4.6 TciObjId

`TciObjId`: ce type est mappé vers l'interface suivante. La représentation Java originelle d'un type TTCN-3 ObjetId se compose d'une séquence ordonnée d'éléments de type `TciObjIdElement`.

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

Méthodes:

- `size()` — Renvoie la longueur de cet identificateur d'objet dans le paramètre TciObjIdElements;
- `setObjElement(TciObjIdElement[] objElements)` — Met cet identificateur d'objet dans la liste des éléments de type objElement;
- `getObjElement(int index)` — Renvoie le paramètre TCIObjIdElement à la position `index`.

8.2.4.7 TciObjIdElement

Un paramètre TciObjIdElement représente un élément d'objet unique à l'intérieur d'une valeur TTCN-3 d'identificateur ObjId. Il peut être réglé au moyen de différentes représentations comme la représentation en caractères ASCII ou comme un nombre entier.

`TciObjIdElement`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `setElementAsAscii(String element)` — Met la représentation interne de cet élément ObjIdElement à la valeur de chaîne `element`;
- `setElementAsNumber(int element)` — Met la représentation interne de cet élément ObjIdElement à la valeur d'entier `element`;
- `getElementAsAscii()` — Renvoie la représentation interne de cet élément ObjIdElement sous forme de chaîne;
- `getElementAsNumber()` — Renvoie la représentation interne de cet élément ObjIdElement sous forme d'entier.

8.2.4.8 CharstringValue

`CharstringValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getString()` — Renvoie la chaîne de la chaîne du type TTCN-3 `charstring`. La représentation textuelle du type TTCN-3 vide `charstring` est `''`, alors que sa longueur est zéro;
- `setString(String value)` — Met ce type `CharstringValue` à la valeur `value`;
- `getChar(int position)` — Renvoie la valeur de caractère de la chaîne de caractères TTCN-3 à la position `position`. La position 0 indique le premier caractère de la chaîne de caractères TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `setChar(int position, char value)` — Met le caractère à la `position` indiquée par la valeur `value`. Les valeurs valides pour la `position` voint de `0` à `length - 1`;
- `getLength()` — Renvoie la longueur de ce type `CharstringValue` en nombre de caractères, renvoie zéro si la valeur de cette chaîne de caractères est `omit`;
- `setLength(int len)` — Met la longueur de ce type `CharstringValue`, exprimée en nombre de caractères, à la valeur `len`.

8.2.4.9 BitstringValue

`BitstringValue`: ce type est mappé vers l'interface suivante:

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

```
    void    setBit (int  position, int value);
    int     getLength ();
    void    setLength (int len);
}
```

Méthodes:

- `getString()` — Renvoie la représentation textuelle de ce type `BitstringValue`, comme défini dans la notation TTCN-3. P. ex., la représentation textuelle de `0101` est `"0101"B`. La représentation textuelle du type TTCN-3 vide `bitstring` est `""B`, alors que sa longueur est zéro;
- `setString(String value)` — Met la valeur de ce type `BitstringValue` conformément à la représentation textuelle définie par la valeur `value`. P. ex., la valeur de ce type `BitstringValue` sera `0101` si la représentation textuelle dans `value` est `"0101"B`;
- `getBit(int position)` — Renvoie la valeur `(0 | 1)` à la `position` de cette chaîne binaire TTCN-3. La position 0 indique le premier bit de la chaîne binaire TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `setBit(int position, int value)` — Met le bit à la position indiquée par la valeur `(0 | 1)`. La position 0 indique le premier bit contenu dans BitstringValue. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `getLength()` — Renvoie la longueur de ce type `BitstringValue` en bits, zéro si la valeur de ce type `BitstringValue` est `omit`;
- `setLength(int len)` — Met la longueur de ce type `BitstringValue`, exprimée en bits, à la valeur `len`.

8.2.4.10 OctetstringValue

`OctetstringValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getString()` — Renvoie la représentation textuelle de ce type `OctetstringValue`, comme défini dans la notation TTCN-3. P. ex., la représentation textuelle de `0xCAFFEE` est `"CAFFEE"O`. La représentation textuelle du type TTCN-3 vide `octetstring` est `""O`, alors que sa longueur est zéro;
- `setString(String value)` — Met la valeur de ce type `OctetstringValue` conformément à la représentation textuelle définie par la valeur value. P. ex., la valeur de ce type `OctetstringValue` sera `0xCAFFEE` si la représentation textuelle contenue dans `value` est `"CAFFEE"O`;
- `getOctet(int position)` — Renvoie la valeur (0.255) à la position de cette chaîne d'octets TTCN-3. La position 0 indique le premier octet de la chaîne d'octets TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `setOctet(int position, int value)` — Met l'octet à la position indiquée par la valeur (0.255). La position 0 indique le premier octet dans la chaîne d'octets. Les valeurs valides pour la position vont de `0` à `length - 1`;

- `getLength()` — Renvoie la longueur de ce type `OctetstringValue` en octets, zéro si la valeur `value` de ce type `OctetstringValue` est `omit`;
- `setLength(int len)` — Met la longueur de ce type `OctetstringValue`, exprimée en octets, à la valeur `len`.

8.2.4.11 UniversalCharstringValue

`UniversalCharstringValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getString()` — Renvoie la représentation textuelle de ce type `UniversalCharstringValue`, comme défini dans la notation TTCN-3;
- `setString(String value)` — Met la valeur de ce type `UniversalCharstringValue` conformément à la représentation textuelle définie par la valeur `value`;
- `getChar(int position)` — Renvoie la valeur `UniversalChar` du type TTCN-3 `universal charstring` à la position indiquée. La position 0 indique le premier caractère universel `UniversalChar` du type TTCN-3 `universal charstring`. Les valeurs valides pour la `position` vont de `0` à `length - 1`;
- `setChar(int position, char value)` — Met le caractère `UniversalChar` à la position indiquée par la valeur `value`. Les valeurs valides pour `position` vont de `0` à `length - 1`;
- `getLength()` — Renvoie la longueur de ce type `UniversalCharstringValue`, exprimée en caractères de type `UniversalChar`, zéro si la valeur de ce type `UniversalCharstringValue` est `omit`;
- `setLength(int len)` — Met la longueur de ce type `UniversalCharstringValue`, exprimée en caractères de type `UniversalChar`, à `len`.

8.2.4.12 HexstringValue

`HexstringValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getString()` — Renvoie la représentation textuelle de ce type `HextstringValue`, comme défini dans la notation TTCN-3. P. ex., la représentation textuelle de `0xAFFEE` est `"AFFEE"H`. La représentation textuelle du type TTCN-3 vide `hexstring` est `""H`, alors que sa longueur est zéro;

- `setString(String value)` — Met la valeur de ce type `HexstringValue` conformément à la représentation textuelle définie par la valeur `value`. P. ex., la valeur de ce type `HexstringValue` sera `0xAFFEE` si la représentation textuelle contenue dans `value` est `"AFFEE"H`;
- `getHex(int position)` — Renvoie la valeur `(0.15)` à la position `position` de cette chaîne hexadécimale TTCN-3. La position 0 indique les premiers caractères hexadécimaux de la chaîne hexadécimale TTCN-3. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `setHex(int position, int value)` — Met le caractère hexadécimal à la position indiquée par la valeur `(0..16)`. La position 0 indique le premier octet dans la chaîne hexadécimale. Les valeurs valides pour la position vont de `0` à `length - 1`;
- `getLength()` — Renvoie la longueur de ce type `HexstringValue` en octets, zéro si la valeur de ce type `HexstringValue` est `omit`;
- `setLength(int len)` — Met la longueur de ce type `HexstringValue`, exprimée en caractères hexadécimaux, à la valeur "len".

8.2.4.13 RecordValue

`RecordValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getField(String fieldName)` — Renvoie la valeur du champ nommé `fieldName`. La valeur de retour est le type de base abstrait commun, car un champ d'enregistrement peut être de tout type défini dans la notation TTCN-3.Si le champ ne peut pas être obtenu à partir de l'enregistrement, la valeur distincte `null` sera renvoyée;
- `setField(String fieldName, Value value)` — Met le champ nommé `fieldName` de l'enregistrement à la valeur `value`. Aucune hypothèse ne doit être effectuée sur la façon dont un champ est mémorisé dans un enregistrement. Une mise en œuvre interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur sera copiée. L'on devrait donc supposer que les modifications subséquentes de la valeur `value` ne seront pas reflétées dans l'enregistrement;
- `getFieldNames()` — Renvoie une liste tabulaire de chaînes de noms de champ et la valeur distincte `null` si l'enregistrement ne contient aucun champ.

8.2.4.14 RecordOfValue

`RecordOfValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getField(String fieldName)` Renvoie la valeur de l'enregistrement `record of` à la position `position` si `position` est une valeur comprise entre zéro et `length -1`; renvoie la valeur distincte `null` sinon. La valeur de retour est le type de base abstrait commun `Value`, car un enregistrement `record of` peut avoir des champs de tout type défini dans la notationTTCN-3;
- `setField(int position, Value value)` Met le champ à la `position` indiquée par la valeur `value`. Si la valeur de `position` est supérieure à `(length -1)` le type `record of` sera étendu de façon à avoir la longueur `(position + 1)`. Les éléments de type `record of` situés entre la position originale `length` et la position `position - 1` seront mis à `OMIT`. Aucune hypothèse ne doit être faite sur la façon dont un champ est mémorisé dans un type `record of`. Une mise en œuvre interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur sera copiée. L'on devrait donc partir du principe que des modifications subséquentes de la valeur `value` ne seront pas reflétées dans le type `record of`;
- `appendField(Value value)` Ajoute la valeur à la fin du type `record of`, c'est-à-dire à la position `length`. Aucune hypothèse ne doit être faite sur la façon dont un champ est mémorisé dans un type `record of`. Une mise en œuvre interne pourrait décider de mémoriser une référence à cette valeur ou de copier la valeur. Il est plus sûr de supposer que la valeur sera copiée. L'on devrait donc supposer que des modifications subséquentes de la valeur `value` ne seront pas reflétées dans le type `record of`;
- `getElementType()` Cette opération renverra le type `Type` des éléments de ce paramètre `record of`;
- `getLength()` Renvoie la longueur réelle de la valeur du type `record of`, zéro si la valeur du type `record of` est `OMIT`;
- `setLength(int len)` Met la longueur du type `record of` à la valeur `len`. Si la longueur `len` est supérieure à la longueur originale, les éléments récemment créés ont la valeur `OMIT`. Si la longueur `len` est inférieure ou égale à la longueur originale, cette opération sera ignorée.

8.2.4.15 UnionValue

`UnionValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getVariant(String variantName)` Renvoie la valeur du nom de réunion TTCN-3 `variantName` si le paramètre `variantName` est égal au résultat de l'opération `getPresentVariantName`, et renvoie la valeur distincte `null` sinon. Le paramètre `variantName` indique le nom de la variante de réunion comme défini dans la notation TTCN-3;
- `setVariant(String variantName, Value value)` Met le nom de la réunion `variantName` à la valeur `value`. Si `variantName` n'est pas défini pour cette réunion, cette opération sera ignorée. Si une autre variante a été sélectionnée, cette nouvelle variante sera sélectionnée à sa place;

- `getPresentVariantName()` — Renvoie le nom de variante qui possède une valeur dans cette réunion, réglée comme étant une chaîne. La valeur distincte `null` sera renvoyée si aucune variante n'est sélectionnée;
- `getVariantNames()` — Renvoie une liste tabulaire de type `String` des noms de variante; renvoie la valeur distincte `null` si la réunion ne contient aucun champ. Si le paramètre `UnionValue` représente l'opération TTCN-3 `anytype`, c'est-à-dire si la classe du type obtenu par l'opération getType() est ANYTYPE, alors tous les types TTCN-3 prédéfinis et définis par l'utilisateur seront renvoyés.

8.2.4.16 EnumeratedValue

`EnumeratedValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getEnum()` — Renvoie l'identificateur de chaîne de ce type `EnumeratedValue`. Cet identificateur est égal à l'identificateur contenu dans la spécification TTCN-3;
- `setEnum(String enumValue)` — Met l'énumération à la valeur `enumValue`. Si `enumValue` n'est pas une valeur autorisée pour cette énumération, l'opération sera ignorée.

8.2.4.17 VerdictValue

`VerdictValue`: ce type est mappé vers l'interface suivante:

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

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

Méthodes:

- `getVerdict()` — Renvoie la valeur d'entier pour cette valeur `VerdictValue`. L'entier est une des constantes suivantes: `ERROR`, `FAIL`, `INCONC`, `NONE`, `PASS`;
- `setVerdict(int verdict)` — Met ce type `VerdictValue` à la valeur `verdict`. Noter qu'un type `VerdictValue` peut être réglé à l'un quelconque des verdicts susmentionnés, à un moment quelconque. Le type `VerdictValue` n'exécute aucun calcul de verdict comme défini dans la notation TTCN-3. Par exemple, il est autorisé de régler le type `VerdictValue` d'abord à `ERROR`, puis à `PASS`.

8.2.4.18 AddressValue

`AddressValue`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getAddress()` Renvoie la valeur représentée par ce type `AddressValue`;
- `setAddress(Value value)` Met ce type `AddressValue` à la valeur `value`.

8.2.5 Mappage de types de journalisation abstraite

Des types additionnels sont définis afin de faciliter la journalisation des concordances entre valeurs et modèles.

8.2.5.1 TciValueTemplate

`TciValueTemplate`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `isOmit()` Renvoie la valeur `true` si le modèle est `omit`, la valeur `false` sinon;
- `isAny()` Renvoie la valeur `true` si le modèle est `any`, la valeur `false` sinon;
- `isAnyOrOmit()` Renvoie la valeur `true` si le modèle est `anyoromit`, la valeur `false` sinon;
- `getTemplateDef()` Cette opération renvoie la définition du modèle.

8.2.5.2 TciNonValueTemplate

`TciNonValueTemplate`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `isAny()` Renvoie la valeur `true` si le modèle est `any`, la valeur `false` sinon;
- `isAll()` Renvoie la valeur `true` si le modèle est `all`, la valeur `false` sinon;
- `getTemplateDef()` Cette opération renvoie la définition du modèle.

8.2.5.3 TciValueList

`TciValueList`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `size()` Renvoie le nombre de valeurs contenues dans cette liste;
- `isEmpty()` Renvoie la valeur `true` si cette liste ne contient aucune valeur;
- `get(int index)` Renvoie la valeur `Value` à la position spécifiée.

8.2.5.4 TciValueDifference

`TciValueDifference`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `getValue()` Renvoie la valeur de ce type `TciValueDifference`;
- `getTciValueTemplate ()` Renvoie le modèle de ce type `TciValueDifference`;
- `getDescription()` Renvoie la description de la discordance.

8.2.5.5 TciValueDifferenceList

`TciValueDifferenceList`: ce type est mappé vers l'interface suivante:

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

Méthodes:

- `size()` Renvoie le nombre de différences contenues dans cette liste;
- `isEmpty()` Renvoie la valeur `true` si cette liste ne contient aucune différence;
- `get(int index)` Renvoie le paramètre `TciValueDifference` à la position spécifiée.

8.3 Constantes

Dans ce mappage vers le langage Java, des constantes ont été spécifiées. Toutes les constantes sont définies comme étant de type `public final static`. Elles sont accessibles à partir de chaque objet et à partir de chaque paquetage. Les constantes définies dans ce paragraphe ne sont pas définies dans le paragraphe sur le langage IDL. En revanche, elles sont le résultat de la spécification des types IDL marqués à l'interface TCI comme étant originels.

Les constantes suivantes peuvent servir à déterminer le mode de communication de paramètres TTCN-3 (voir également le § 8.2.2.2).

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

Pour la valeur value paramétrique distincte `null`, la valeur du paramètre codé doit être réglée à la valeur Java `null`.

Les constantes suivantes doivent servir à la manipulation des valeurs (voir également le § 8.2.2.4).

- `org.etsi.ttcn.tci.TciTypeClass.ADDRESS`;
- `org.etsi.ttcn.tci.TciTypeClass.ANYTYPE`;
- `org.etsi.ttcn.tci.TciTypeClass.BITSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.BOOLEAN`;
- `org.etsi.ttcn.tci.TciTypeClass.CHARSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.COMPONENT`;
- `org.etsi.ttcn.tci.TciTypeClass.ENUMERATED`;
- `org.etsi.ttcn.tci.TciTypeClass.FLOAT`;
- `org.etsi.ttcn.tci.TciTypeClass.HEXSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.INTEGER`;

- `org.etsi.ttcn.tci.TciTypeClass.OBJID`;
- `org.etsi.ttcn.tci.TciTypeClass.OCTETSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.RECORD`;
- `org.etsi.ttcn.tci.TciTypeClass.RECORD_OF`;
- `org.etsi.ttcn.tci.TciTypeClass.SET`;
- `org.etsi.ttcn.tci.TciTypeClass.SET_OF`;
- `org.etsi.ttcn.tci.TciTypeClass.UNION`;
- `org.etsi.ttcn.tci.TciTypeClass.UNIVERSAL_CHARSTRING`;
- `org.etsi.ttcn.tci.TciTypeClass.VERDICT`.

Les constantes suivantes doivent servir au traitement de composant (voir également le § 8.2.2.5).

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

8.4 Mappage d'interfaces

La définition de l'interface TCI en langage IDL détermine quatre interfaces: **`TCI-TM`**, **`TCI-CH`**, **`TCI-CD`** et **`TCI-TL`**. Les opérations sont définies dans différents sens à l'intérieur de cette interface, c'est-à-dire que certaines opérations ne peuvent être appelées que par l'exécutable TTCN-3 (TE), que par l'adaptateur de système (SA) ou que par l'adaptateur de plate-forme (PA) dans la gestion et commande de test (TMC) alors que d'autres opérations ne peuvent être appelées que par la gestion TMC dans l'exécutable TE. Cette différence se retrouve dans la subdivision des interfaces TCI du langage IDL en deux sous-interfaces, chacune étant assortie du suffixe `Required` ou `Provided`.

Tableau 3/Z.145 – Sous-interfaces

Appelant/Appelé	TE	TM	CD	CH	SA	PA	TL
TE	–	TCI-TM fournie	TCI-CD fournie	TCI-CH fournie	Voir Rec. UIT-T Z.144 [1]	Voir Rec. UIT-T Z.144 [1]	TCI-TL fournie
TM	TCI-TM requise	–	–	–	–	–	TCI-TL fournie
CD	TCI-CD requise	–	–	–	–	–	TCI-TL fournie
CH	TCI-CH requise	–	–	–	–	–	TCI-TL fournie
SA	Voir Rec. UIT-T Z.144	–	–	–	–	–	TCI-TL fournie
PA	Voir Rec. UIT-T Z.144	–	–	–	–	–	TCI-TL fournie
TL	–	–	–	–	–	–	–

Toutes les méthodes définies dans ces interfaces devraient présenter le comportement défini dans le paragraphe 7.

8.4.1 L'interface TCI-TM

L'interface `TCI-TM` est subdivisée en deux sous-interfaces: la sous-interface `TCI-TM Provided`, qui définit les appels allant de l'exécutable TE à la gestion TM et la sous-interface `TCI-TM Required`, qui définit les appels allant de la gestion TM à l'exécutable TE.

8.4.1.1 Sous-interface TCI-TM fournie

La sous-interface `TCI-TM Provided` est mappée vers l'interface suivante:

```
// TCI-TM
// TE -> TM
package org.etsi.ttcn.tci;
public interface TciTMProvided {
    public void tciTestCaseStarted(TciTestCaseId testCaseId, TciParameterList parameterList, Float
    timer);
```

```
    public void     tciTestCaseTerminated( VerdictValue verdict, TciParameterList parameterList);
    public void     tciControlTerminated();
    public Value    tciGetModulePar(TciModuleParameterId parameterId);
    public void     tciLog(TriComponentId testComponentId, String message);
    public void     tciError(String message);
}
```

8.4.1.2 Sous-interface TCI-TM requise

La sous-interface `TCI-TM Required` est mappée vers l'interface suivante:

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

8.4.2 L'interface TCI-CD

L'interface `TCI-CD` est subdivisée en deux sous-interfaces: la sous-interface `TCI-CD Provided`, qui définit les appels allant de l'exécutable TE au codec CD et la sous-interface `TCI-CD Required`, qui définit les appels allant du codec CD à l'exécutable TE.

8.4.2.1 Sous-interface TCI-CD fournie

La sous-interface `TCI-CD Provided` est mappée vers l'interface suivante:

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

8.4.2.2 Sous-interface TCI-CD requise

La sous-interface `TCI-CD Required` est mappée vers l'interface suivante:

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

8.4.3 L'interface TCI-CH

L'interface `TCI-CH` est subdivisée en deux sous-interfaces: la sous-interface `TCI-CH Provided`, qui définit les appels allant de l'exécutable TE au dispositif CH et la sous-interface `TCI-CH Required`, qui définit les appels allant du dispositif CH à l'exécutable TE.

8.4.3.1 Sous-interface TCI-CH fournie

La sous-interface `TCI-CH Provided` est mappée vers l'interface suivante:

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

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

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

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

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

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

    public void     tciStopTestComponentReq(TriComponentId comp) ;

    public void     tciConnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void     tciDisconnectReq(TriPortId fromPort, TriPortId toPort) ;

    public void     tciTestComponentTerminatedReq(TriComponentId comp, VerdictValue verdict) ;

    public boolean  tciTestComponentRunningReq(TriComponentId comp) ;

    public TriComponentId   tciGetMTCReq() ;

    public void     tciMapReq(TriPortId fromPort, TriPortId toPort);

    public void     tciUnmapReq(TriPortId fromPort, TriPortId toPort);

    public void     tciExecuteTestCaseReq(TriComponentId testComponentId,
                                          TriPortIdList tsiPortList);

    public void     tciResetReq() ;

    public boolean  tciTestComponentDoneReq(TriComponentId testComponentId) ;
```

```
    public void     tciKillTestComponentReq(TriComponentId component)

    public boolean  tciTestComponentAliveReq(TriComponentId component)

    public boolean  tciTestComponentKilledReq(TriComponentId component)
}
```

8.4.3.2 Sous-interface TCI-CH requise

La sous-interface `TCI-CH Required` est mappée vers l'interface suivante:

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

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

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

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

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

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

    public void     tciStopTestComponent(TriComponentId comp) ;

    public void     tciConnect(TriPortId fromPort, TriPortId toPort) ;

    public void     tciDisconnect(TriPortId fromPort, TriPortId toPort) ;

    public void     tciTestComponentTerminated(TriComponentId comp, VerdictValue verdict);

    public boolean  tciTestComponentRunning(TriComponentId comp);

    public boolean  tciTestComponentDone(TriComponentId comp);

    public TriComponentId   tciGetMTC();

    public void     tciExecuteTestCase(TciTestCaseId TestCaseId, TriPortIdList tsiPortList);

    public void     tciReset();

    public void     tciMap(TriPortId fromPort, TriPortId toPort);

    public void     tciUnmap(TriPortId fromPort, TriPortId toPort);

    public void     tciKillTestComponent(TriComponentId component)

    public boolean  tciTestComponentAlive(TriComponentId component)

    public boolean  tciTestComponentKilled(TriComponentId component)
}
```

8.4.4 L'interface TCI-TL

L'interface `TCI-TL` contient seulement la sous-interface `TCI-TL Provided`, qui définit les appels allant vers la journalisation TL à partir de l'exécutable TE, de la gestion TM, du dispositif CH, du codec CD, et des adaptateurs SA et PA.

8.4.4.1 Sous-interface TCI-TL fournie

La sous-interface `TCI-TL Provided` est mappée vers l'interface suivante:

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

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

```
        TriPortId port, TriSignatureId signature, TciParameterList parsValue, Value excValue,
        TriComponentIdList toList, TriStatus transmissionFailure);
public void tliPrCatchDetected_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TriException exc, TriAddress address);
public void tliPrCatchDetected_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, Value excValue, TriComponentId from);
public void tliPrCatchMismatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,  TciValueDifferenceList diffs,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrCatchMismatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,  TciValueDifferenceList diffs,
        TriComponentId from, TciNonValueTemplate fromTmpl);
public void tliPrCatch_m(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,
        TriAddress address, TciValueTemplate addressTmpl);
public void tliPrCatch_c(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue,
        Value excValue, TciValueTemplate excTmpl,
        TriComponentId from, TciNonValueTemplate fromTmpl);
public void tliPrCatchTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature);
public void tliPrCatchTimeout(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port, TriSignatureId signature, TciParameterList parsValue);
public void tliCCreate(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, String name);
public void tliCStart(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TciBehaviourId name, TciParameterList parsValue);
public void tliCRunning(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TBoolean status);
public void tliCAlive(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp, TBoolean status);
public void tliCStop(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp);
public void tliCKill(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId comp);
public void tliCDoneMismatch(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate compTmpl);
public void tliCDone(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate compTmpl);
public void tliCKilledMismatch(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate compTmpl);
public void tliCKilled(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate compTmpl);
public void tliCTerminated(String am, int ts, String src, int line, TriComponentId c,
        VerdictValue verdict);
public void tliPConnect(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
public void tliPDisconnect(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
public void tliPMap(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
public void tliPUnmap(String am, int ts, String src, int line, TriComponentId c,
        TriComponentId c1, TriPortId port1, TriComponentId c2, TriPortId port2);
public void tliPClear(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
public void tliPStart(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
public void tliPStop(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
public void tliPHalt(String am, int ts, String src, int line, TriComponentId c,
        TriPortId port);
public void tliEncode(String am, int ts, String src, int line, TriComponentId c,
        Value val, TriStatus encoderFailure, TriMessage msg, String codec);
public void tliDecode(String am, int ts, String src, int line, TriComponentId c,
        TriMessage msg, TriStatus decoderFailure, Value val, String codec);
public void tliTTimeoutDetected(String am, int ts, String src, int line, TriComponentId c,
        TriTimerId timer);
public void tliTTimeoutMismatch(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate timerTmpl);
public void tliTTimeout(String am, int ts, String src, int line, TriComponentId c,
        TciNonValueTemplate timerTmpl);
public void tliTStart(String am, int ts, String src, int line, TriComponentId c,
        TriTimerId timer, TriTimerDuration dur);
public void tliTStop(String am, int ts, String src, int line, TriComponentId c,
        TriTimerId timer);
public void tliTRead(String am, int ts, String src, int line, TriComponentId c,
        TriTimerId timer, TriTimerDuration elapsed);
```

```
    public void tliTRunning(String am, int ts, String src, int line, TriComponentId c,
            TriTimerId timer, TBoolean status);
    public void tliSEnter(String am, int ts, String src, int line, TriComponentId c,
            String name, TciParameterList parsValue, String kind);
    public void tliSLeave(String am, int ts, String src, int line, TriComponentId c,
            String name, Value returnValue, String kind);
    public void tliVar(String am, int ts, String src, int line, TriComponentId c,
            String name, Value varValue);
    public void tliModulePar(String am, int ts, String src, int line, TriComponentId c,
            String name, Value parValue);
    public void tliGetVerdict(String am, int ts, String src, int line, TriComponentId c,
            VerdictValue verdict);
    public void tliSetVerdict(String am, int ts, String src, int line, TriComponentId c,
            VerdictValue verdict);
    public void tliLog(String am, int ts, String src, int line, TriComponentId c,
            TciValueList log);
    public void tliAEnter(String am, int ts, String src, int line, TriComponentId c);
    public void tliALeave(String am, int ts, String src, int line, TriComponentId c);
    public void tliADefaults(String am, int ts, String src, int line, TriComponentId c);
    public void tliAActivate(String am, int ts, String src, int line, TriComponentId c,
            String name, TciParameterList pars, Value ref);
    public void tliADeactivate(String am, int ts, String src, int line, TriComponentId c,
            Value ref);
    public void tliANomatch(String am, int ts, String src, int line, TriComponentId c);
    public void tliARepeat(String am, int ts, String src, int line, TriComponentId c);
    public void tliAWait(String am, int ts, String src, int line, TriComponentId c);
}
```

8.5 Paramètres facultatifs

Le paragraphe 7.3 définit qu'une valeur réservée doit servir à indiquer l'absence d'un paramètre facultatif. Pour le mappage vers le langage Java, la valeur Java `null` doit servir à indiquer l'absence d'une valeur facultative. Par exemple si, dans l'opération `tciReplyConnected`, le paramètre de valeur doit être omis, l'invocation de l'opération doit être: `tciReplyConnected (sender, receiver, signature, parameterList, null)`.

8.6 Initialisation d'interface TCI

Toutes les méthodes sont non statiques, c'est-à-dire que les opérations ne peuvent être appelées que sur des objets. Comme la présente Recommandation ne définit pas de stratégies concrètes d'implémentation des dispositifs TE, TM, CD et CH. Le mécanisme permettant à l'exécutable TE, à la gestion TM, au codec CD ou au dispositif CH d'arriver à connaître les pointeurs vers les objets correspondants est hors du domaine d'application de la présente Recommandation.

Les vendeurs d'utilitaires doivent offrir, aux développeurs des dispositifs TM, CD et CH, des méthodes permettant d'enregistrer les dispositifs TE, TM, CD et CH auprès de leurs partenaires de communication respectifs.

8.7 Traitement des erreurs

Toutes les opérations appelées à partir du dispositif TM,CH ou CD, qui renvoient une valeur, ont réussi. Si une situation erronée a été identifiée par l'exécutable TE, une erreur de test élémentaire sera communiquée à l'utilisateur au moyen des procédures définies dans le § 7.3.1.2.5 (`tciError`). Si une opération appelée par l'exécutable TE dans le dispositif TM, CH, CD, ou TL produit une erreur, cette situation erronée devrait être communiquée à l'exécutable TE au moyen des procédures définies dans le § 7.3.2.1.12 (`tciErrorReq`).

En dehors de ce traitement des erreurs, aucun autre traitement des erreurs n'est défini dans le cadre de ce mappage vers le langage Java. En particulier, aucun mécanisme de traitement des exceptions n'est défini.

9 Mappage vers le langage ANSI-C

9.1 Introduction

Le présent paragraphe définit le mappage du langage ANSI-C à l'interface TRI pour les données d'interface TCI spécifiées dans le § 7.2 et pour les opérations d'interface TCI spécifiées dans le § 7.3.

9.2 Interfaces avec une valeur

Interface TCI-IDL	Représentation en ANSI-C	Notes et commentaires
Type		
TciModuleIdType getDefiningModule()	`TciModuleIdType tciGetDefiningModule(TciType inst)`	
TString getName()	`String tciGetName(TciType inst)`	Type "chaîne" réutilisé à partir du langage IDL (recommandation de l'OMG)
TciTypeClassType getTypeClass()	`TciTypeClassType tciGetTypeClass(TciType inst)`	
Value newInstance()	`TciValue tciNewInstance(TciType inst)`	
TString getTypeEncoding()	`String tciGetTypeEncoding(TciType inst)`	
TStringSeq getTypeExtension()	`String* getTypeExtension(TciType inst)`	
TString getTypeEncodingVariant()	`String tciGetTypeEncodingVariant(TciType inst)`	
Value		
TString getValueEncoding()	`String tciGetValueEncoding(TciValue inst)`	
TString getValueEncodingVariant()	`String tciGetValueEncodingVariant(TciValue inst)`	
Type getType()	`TciType tciGetType(TciValue inst)`	
TBoolean notPresent()	`Boolean tciNotPresent(TciValue inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
IntegerValue		
TInteger getInt()	`String tciGetIntAbs(TciValue inst)`	Renvoie la valeur absolue de l'entier (en base décimale) sous forme d'une chaîne de caractères ASCII.
	`unsigned long int tciGetIntNumberOfDigits (TciValue inst)`	Renvoie le nombre de caractères contenus dans une valeur d'entier.
	`Boolean tciGetIntSign(TciValue inst)`	Renvoie la valeur true si le nombre est positif, la valeur false sinon.
	`char tciGetIntDigit (TciValue inst, unsigned long int position)`	Renvoie la valeur du caractère situé à la position 'position', où "position 0" est le caractère de poids faible.
void setInt(in TInteger value)	`void tciSetIntAbs(TciValue inst, String value)`	Met la valeur absolue (en base décimale) de l'entier sous forme d'une chaîne de caractères ASCII. Le premier chiffre ne doit pas être 0 à moins que la valeur ne soit 0.
	`void tciSetIntNumberOfDigits (TciValue inst, unsigned long int dig_num)`	Met le nombre de caractères numériques dans une valeur d'entier.
	`void tciSetIntSign (TciValue inst, Boolean sign)`	Met le sign à + (true) ou – (false).
	`void tciSetIntDigit (TciValue inst, unsigned long int position, char digit)`	Met la valeur du caractère à la position 'position', où "position 0" est le caractère de poids faible.
FloatValue		
TFloat getFloat()	`double tciGetFloatValue(TciValue inst)`	

Interface TCI-IDL	Représentation en ANSI-C	Notes et commentaires
void setFloat (in TFloat value)	`void tciSetFloatValue(TciValue inst, double value)`	
BooleanValue		
TBoolean getBoolean()	`Boolean tciGetBooleanValue(TciValue inst)`	
void setBoolean (in TBoolean value)	`void tciSetBooleanValue (TciValue inst, Boolean value)`	
ObjidValue		
TObjid getObjid()	`TciObjidValue tciGetTciObjidValue(TciValue inst)`	
void setObjid(in TObjid value)	`void tciSetObjidValue(TciValue inst, TciObjidValue value)`	
CharstringValue		
TString getString()	`TciCharStringValue tciGetCStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetCStringValue (TciValue inst, TciCharString value)`	
TChar getChar (in TInteger position)	`char tciGetCStringCharValue (TciValue inst, long int position)`	
void setChar (in TInteger position, in char value)	`void tciSetCStringCharValue (TciValue inst, long int position, char value)`	
TInteger getLength()	`unsigned long int tciGetCStringLength (TciValue inst)`	
void setLength(in TInteger len)	`void tciSetCStringLength (TciValue inst, unsigned long int len)`	
UniversalCharstringValue		
TString getString()	`TciUCStringValuetciGetUCStringValue(TciValue inst)`	
void setString (in TString value)	`void tciSetUCStringValue (TciValue inst, TciUCStringValue value)`	
TUniversalChar getChar (in TInteger position)	`void tciGetUCStringCharValue (TciValue inst, unsigned long int position, TciUCValue result)`	
void setChar (in TInteger position, in TUniversalChar value)	`void tciSetUCStringCharValue (TciValue inst, unsigned long int position, TciUCValue value)`	
TInteger getLength()	`unsigned long int tciGetUCStringLength(TciValue inst)`	
void setLength (in TInteger len)	`void tciSetUCStringLength (TciValue inst, unsigned long int len)`	
BitstringValue		
TString getString()	`String tciGetBStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetBStringValue (TciValue inst, String value)`	
TChar getBit (in integer position)	`int tciGetBStringBitValue (TciValue inst, long int position)`	
void setBit (in TInteger position, in TInteger value)	`void tciSetBStringBitValue (TciValue inst, unsigned long int position, int value)`	
TInteger getLength()	`unsigned long int tciGetBStringLength(TciValue inst)`	
void setLength (in TInteger len)	`void tciSetBStringLength (TciValue inst, long int len)`	
HexstringValue		
TString getString()	`String tciGetHStringValue(TciValue inst)`	

Interface TCI-IDL	Représentation en ANSI-C	Notes et commentaires
void setString (in TString value)	`void tciSetHStringValue (TciValue inst, String value)`	
TChar getHex (in TInteger position)	`int tciGetHStringHexValue (TciValue inst, unsigned long int position)`	
void setBit (in TInteger position, in TInteger value)	`void tciSetHStringHexValue (TciValue inst, unsigned long int position, int value)`	
TInteger getLength()	`long int tciGetHStringLength(TciValue inst)`	
void setLength(in TInteger len)	`void tciSetHStringLength (TciValue inst, unsigned long int len)`	
OctetstringValue		
TString getString()	`String tciGetOStringValue(TciValue inst)`	
void setString(in TString value)	`void tciSetOStringValue (TciValue inst, String value)`	
TChar getOctet(in TInteger position)	`int tciGetOStringOctetValue (TciValue inst, unsigned long int position)`	
void setOctet (in TInteger position, in TInteger value)	`void tciSetOStringOctetValue (TciValue inst, unsigned long int position, int value)`	
TInteger getLength()	`unsigned long int tciGetOStringLength(TciValue inst)`	
void setLength(in TInteger len)	`void tciSetOStringLength (TciValue inst, unsigned long int len)`	
RecordValue		
Value getField (in TString fieldName)	`TciValue tciGetRecFieldValue (TciValue inst, String fieldName)`	
void setField (in TString fieldName, in Value value)	`void tciSetRecFieldValue (TciValue inst, String fieldName, TciValue value)`	
TString[] getFieldNames()	`char** tciGetRecFieldNames(TciValue inst)`	Renvoie une liste tabulaire des noms de champ terminée par NULL.
RecordOfValue		
Value getField (in TInteger position)	`TciValue tciGetRecOfFieldValue (TciValue inst, unsigned long int position)`	
void setField (in TInteger position, in Value value)	`void tciSetRecOfFieldValue (TciValue inst, unsigned long int position, TciValue value)`	
void appendField (in Value value)	`void tciAppendRecOfFieldValue (TciValue inst, TciValue value)`	
Type getElementType()	`TciType tciGetRecOfElementType(TciValue inst)`	
TInteger getLength()	`unsigned long int tciGetRecOfLength(TciValue inst)`	
void setLength (in TInteger len)	`void tciSetRecOfLength (TciValue inst, unsigned long int len)`	
UnionValue		
Value getVariant (in TString variantName)	`TciValue tciGetUnionVariant (TciValue inst, String variantName)`	
void setVariant (in TString variantName, in Value value)	`void tciSetUnionVariant (TciValue inst, String variantName, TciValue value)`	

Interface TCI-IDL	Représentation en ANSI-C	Notes et commentaires
TString getPresentVariantName()	`String tciGetUnionPresentVariantName (TciValue inst)`	
TString[] getVariantNames()	`char** tciGetUnionVariantNames(TciValue inst)`	Renvoie une liste tabulaire des noms de champ terminée par NULL.
EnumeratedValue		
TString getEnum()	`String tciGetEnumValue(TciValue inst)`	
void setEnum (in TString enumValue)	`void tciSetEnumValue (TciValue inst, String enumValue)`	
VerdictValue		
TInteger getVerdict()	`int tciGetVerdictValue(TciValue inst)`	
void setVerdict (in TInteger verdict)	`void tciSetVerdictValue(TciValue inst, int verdict)`	
AddressValue		
Value getAddress()	`TciValue tciGetAddressValue(TciValue inst)`	
void setAddress (in Value value)	`void tciSetAddressValue(TciValue inst, TciValue value)`	

9.3 Interface avec une journalisation

Interface TCI-IDL	Représentation en ANSI-C	Notes et commentaires
TciValueTemplate		
TBoolean isOmit()	`Boolean tciIsOmit(TciValueTemplate inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
TBoolean isAny()	`Boolean tciIsAny(TciValueTemplate inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
TBoolean isAnyOrOmit()	`Boolean tciIsAnyOrOmit(TciValueTemplate inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
TString getTemplateDef()	`String tciGetTemplateDef(TciValueTemplate inst)`	Type "chaîne" réutilisé à partir du langage IDL (recommandation de l'OMG)
TciNonValueTemplate		
TBoolean isAny()	`Boolean tciIsAny(TciNonValueTemplate inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
TBoolean isAll()	`Boolean tciIsAll(TciNonValueTemplate inst)`	Type "booléen" réutilisé à partir du langage IDL (recommandation de l'OMG)
TString getTemplateDef()	`String tciGetTemplateDef(TciNonValueTemplate inst)`	Type "chaîne" réutilisé à partir du langage IDL (recommandation de l'OMG)

9.4 Interfaces avec une opération

Sous-interface TCI-TM requise		
void tciRootModule (in TciModuleIdType moduleId)	`void tciRootModule(String moduleId)`	
TciModuleParameterListType tciGetModuleParameters (in TciModuleIdType moduleName)	`TciModuleParameterListType tciGetModuleParameters (TciModuleIdType moduleName)`	
TciTestCaseIdListType tciGetTestCases()	`TciTestCaseIdListType tciGetTestCases()`	

TciParameterTypeListType
tciGetTestCaseParameters
(in TciTestCaseIdType testCaseId)
	`TciParameterTypeListType`
	`tciGetTestCaseParameters`
	`(TciTestCaseIdType testCaseId)`	

TriPortIdListType tciGetTestCaseTSI (in TciTestCaseIdType testCaseId)	`TriPortIdList tciGetTestCaseTSI` `(TciTestCaseIdType testCaseId)`	

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

void tciStopTestCase()	`void tciStopTestCase()`	
TriComponentId tciStartControl()	`TriComponentId tciStartControl()`	
void tciStopControl()	`void tciStopControl()`	
Sous-interface TCI-TM requise		

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

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

void tciControlTerminated()	`void tciControlTerminated()`	

Value tciGetModulePar
(in TciModuleParameterIdType parameterId)
	`tciValue tciGetModulePar`
	`(TciModuleParameterIdType`
	`parameterId)`	

void tciLog(in TString message)	`void tciLog(String message)`	
void tciError(in TString message)	`void tciError(String message)`	
Sous-interface TCI-CD requise		
Type getTypeForName (in String typeName)	`TciType tciGetTypeForName` `(String typeName)`	
Type getInteger()	`TciType tciGetIntegerType()`	
Type getFloat()	`TciType tciGetFloatType()`	
Type getBoolean()	`TciType tciGetBooleanType()`	
Type getChar()	`TciType tciGetCharType()`	
Type getUniversalChar()	`TciType tciGetUniversalCharType()`	
Type getObjid()	`TciType tciGetTciObjidType()`	
Type getCharstring()	`TciType tciGetTciCharstringType()`	
Type getUniversalCharstring()	`TciType` `tciGetUniversalCharstringType()`	
Type getHexstring()	`TciType tciGetHexstringType()`	
Type getBitstring()	`TciType tciGetBitstringType()`	
Type getOctetstring()	`TciType tciGetOctetstringType()`	
Type getVerdict()	`TciType tciGetVerdictType()`	
void tciErrorReq(in String message)	`void tciErrorReq(String message)`	
Sous-interface TCI-CD fournie		

Value decode
(in TriMessageType message,
in Type decodingHypothesis)
	`TciValue tciDecode`
	`(BinaryString message,`
	`TciType decHypothesis)`
		Type "chaîne binaire" réutilisé à partir de TRI

TriMessageType encode (in Value value)	`BinaryString tciEncode(TciValue value)`	
Sous-interface TCI-CH requise		

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.5 Données

TCI IDL ADT	Représentation en ANSI-C (Définition de type)	Notes et commentaires
TciModuleIdType	**QualifiedName**	
TciModuleParameterType	typedef struct TciModuleParameterType { String parName; TciValue defaultValue; } TciModuleParameterType;	
TciModuleParameterListType	typedef struct TciModuleParameterListType { long int length; TciModuleParameterType *modParList; } TciModuleParameterListType;	

TCI IDL ADT	Représentation en ANSI-C (Définition de type)	Notes et commentaires
TciModuleIdType	**QualifiedName**	

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

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

TciParameterListType	typedef struct TciParameterListType
	{
	  long int length;
	  TciParameterType *parList;
	} TciParameterListType;
		Une longueur 0 doit être interprétée comme étant une "liste vide".

TciParameterTypeListType	typedef struct TciParameterTypeListType
	{
	  long int length;
	  TciType *parList;
	} TciParameterTypeListType;
		Une longueur 0 doit être interprétée comme étant une "liste vide".

TciTestCaseIdListType	typedef struct TciTestCaseIdListType
	{
	  long int length;
	  QualifiedName *idList;
	} TciTestCaseIdListType;
		Une longueur 0 doit être interprétée comme étant une "liste vide".

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

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

TciBehaviourIdType	QualifiedName	

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

TciValueDifferenceList	typedef struct TciValueDifferenceList
	{
	  long int length;
	  TciValueDifference[] diffList;
	} TciValueDifferenceList;
		Une longueur 0 doit être interprétée comme étant une "liste vide".

9.6 Considérations diverses

TCI concept	Représentation en ANSI-C	Notes et commentaires
Verdict representation		
NONE	`const int TCI_VERDICT_NONE = 0`	Etant donné que l'interface TciVerdictValue est définie en termes d'entiers, un accord doit être trouvé quant à la valeur qui définit chaque verdict.
PASS	`const int TCI_VERDICT_PASS = 1`	
INCONC	`const int TCI_VERDICT_INCONC = 2`	
FAIL	`const int TCI_VERDICT_FAIL = 3`	
ERROR	`const int TCI_VERDICT_ERROR = 4`	
Objid representation		
Objid	`typedef struct TciObjidValue { long int length; TciObjidElem *elements; } TciObjidValue;`	Etant donné que la valeur d'identificateur d'objet est renvoyée "en l'état" au moyen de l'interface avec la valeur d'identificateur d'objet, une représentation doit être définie.
TciObjidElem	`typedef struct TciObjidElemValue { char* elem_as_ascii; long int elem_as_number; void* aux; } TciObjidElemValue;`	
CharstringValue representation		
TciCharString	`typedef struct TciCharStringValue { unsigned long int length; char* string; } TciCharStringValue`	
Universal Character[string] representation		
Universal Char	`typedef unsigned char[4] TciUCValue`	
Universal Charstring	`typedef struct TciUCStringValue { unsigned long int length; TciUCType *string; } TciUCStringValue;`	

10 Mappage vers le langage XML du groupe W3C

10.1 Introduction

Le présent paragraphe introduit le mappage à l'interface TCI vers le langage XML [5], [6] et [7] pour l'interface de journalisation d'interface TCI. Le mappage vers le langage XML pour l'interface de journalisation définit comment les définitions du langage IDL, décrites dans le § 7, sont mappées vers le langage XML. Les définitions de schéma pour ce mappage sont indiquées dans l'Annexe B.

10.2 Portées

Le module IDL **`tciInterface`** est mappé vers un schéma en langage XML dans l'espace nominatif suivant: `http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI`.

Ce schéma utilise les autres schémas suivants:

- http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes pour le mappage de types simples vers le langage XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Types pour le mappage de types structurés vers le langage XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Values pour le mappage de valeurs vers le langage XML.
- http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates pour le mappage de modèles vers le langage XML.

- http://uri.etsi.org/ttcn-3/3.0.0/tci/Events pour le mappage d'événements de journalisation vers le langage XML.

10.3 Mappage de type

10.3.1 Mappage de types simples

10.3.1.1 TBoolean

Le type IDL **`TBoolean`** est mappé vers le type de base xsd `boolean`.

10.3.1.2 TString

Le type IDL **`TString`** est mappé vers le type de base xsd `string`.

10.3.1.3 TInteger

Le type IDL **`TInteger`** est mappé vers le type de base xsd `integer`.

10.3.1.4 TriTimerDurationType

Le type IDL **`TriTimerDurationType`** est mappé vers le type de base xsd `float`.

10.3.1.5 TciParameterPassingModeType

Le type IDL **`TciParameterPassingModeType`** est mappé vers le type de base xsd `string` avec les valeurs d'énumération `'in'`, `'out'` et `'inout'`.

10.3.1.6 TriStatusType

Le type IDL **`TriStatusType`** est mappé vers le type de base xsd `string` avec les valeurs d'énumération `'TRI_Ok'` et `'TRI_Error'`.

10.3.1.7 TciStatusType

Le type IDL **`TriStatusType`** est mappé vers le type de base xsd `string` avec les valeurs d'énumération `'TCI_Ok'` et `'TCI_Error'`.

10.3.2 Mappage de type complexe

10.3.2.1 TriPortIdType

`TriPortIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `comp` L'identificateur de composant à l'interface TRI;
- `port` L'identification du port.

Attributs:

- néant

10.3.2.2 TriComponentIdType

`TriComponentIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `id` L'identificateur du composant à l'interface TRI;
- `null` L'identificateur `null`, à utiliser s'il n'y a aucun identificateur de composant à l'interface TRI.

Attributs:

- néant.

10.3.2.3 TriComponentIdListType

`TriComponentIdListType` est mappé vers le type complexe suivant:

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

Eléments:

- `comp` Les identificateurs de composants à l'interface TRI contenus dans cette liste.

Attributs:

- néant.

10.3.2.4 Port

`Port` est mappé vers le type complexe suivant:

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

Eléments:

- `id` L'identificateur de port;
- `port` L'indice de port.

Attributs:

- néant.

10.3.2.5 Id

`Id` est utilisé comme identification de composants, ports et temporisateurs. Il est mappé vers le type complexe suivant:

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

Eléments:

- `name` Nom du composant, port ou temporisateur;
- `id` Représentation interne du composant, port ou temporisateur;
- `type` Type du composant, port ou temporisateur.

Attributs:

- néant.

10.3.2.6 TriMessageType

`TriMessageType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` Message codé.

Attributs:

- néant.

10.3.2.7 TriParameterType

`TriParameterType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` Le paramètre codé.

Attributs:

- `name` Nom du paramètre;
- `mode` Le mode de communication de paramètre.

10.3.2.8 TriParameterListType

`TriParameterListType` est mappé vers le type complexe suivant:

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

Eléments:

- paramètres contenus dans cette liste.

Attributs:

- néant.

10.3.2.9 TriAddressType

`TriAddresseType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` La valeur d'adresse.

Attributs:

- néant.

10.3.2.10 TriAddressListType

`TriAddressListType` est mappé vers le type complexe suivant:

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

Eléments:

- `addr` Les destinataires contenus dans cette liste.

Attributs:

- néant.

10.3.2.11 TriExceptionType

`TriExceptionType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` L'exception.

Attributs:

- néant.

10.3.2.12 TriSignatureIdType

`TriSignatureIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` La signature.

Attributs:

- néant.

10.3.2.13 TriAddressType

`TriAddresseType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` L'adresse dans le système SUT.

Attributs:

- néant.

10.3.2.14 TriTimerIdType

`TriTimerIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `id` L'identification de temporisateur.

Attributs:

- néant.

10.3.2.15 TriTimerDurationType

`TriTimerDurationType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` Durée de la temporisation.

Attributs:

- néant.

10.3.2.16 QualifiedName

`QualifiedName` sert à qualifier entièrement des paramètres de module, des variables, etc. Ce type est mappé vers le type complexe suivant:

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

Eléments:

- `moduleName` Le nom de module du module TTCN-3.
- `baseName` Nom de l'objet qui est entièrement qualifié.

Attributs:

- néant.

10.3.2.17 TciBehaviourIdType

`TciBehaviourIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `name` Le nom qualifié du comportement.

Attributs:

- néant.

10.3.2.18 TciTestCaseIdType

`TciTestCaseIdType` est mappé vers le type complexe suivant:

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

Eléments:

- `name` Le nom qualifié de test élémentaire.

Attributs:

- néant.

10.3.2.19 TciParameterType

`TciParameterType` est mappé vers le type complexe suivant:

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

Eléments:

- `val` Le paramètre codé.

Attributs:

- `name` Nom du paramètre.
- `mode` Le mode de communication de paramètre.

10.3.2.20 TciParameterListType

`TciParameterListType` est mappé vers le type complexe suivant:

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

Eléments:

- `par` Paramètres contenus dans cette liste.

Attributs:

- néant.

10.3.3 Mappage de valeur abstraite

10.3.3.1 Value

`Value` est mappé vers le type complexe suivant:

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

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

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

Choix d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- `name` Nom de la valeur, si connu.
- `type` Type de la valeur, si connu.
- `module` Module de la valeur, si connu.

10.3.3.2 IntegerValue

`IntegerValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur d'entier sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de "Value".

10.3.3.3 FloatValue

`FloatValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur en virgule flottante sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.4 BooleanValue

`BooleanValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur booléenne sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.5 ObjidValue

`ObjidValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur d'identificateur d'objet sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.6 VerdictValue

`VerdictValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` Valeur de verdict sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.7 BitstringValue

`BitstringValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur de chaîne binaire sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.8 HexstringValue

`HexstringValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur de chaîne hexadécimale sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.9 OctetstringValue

`OctetstringValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur de chaîne d'octets sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.10 CharstringValue

`CharstringValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur de chaîne de caractères sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.11 UniversalCharstringValue

`UniversalCharstringValue` est mappé vers le type complexe suivant:

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

Simple Content:

- `base` La valeur de chaîne de caractères universels sous forme de chaîne.
- `extension` Les mêmes attributs que ceux de Value.

10.3.3.12 RecordValue

`RecordValue` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.13 RecordOfValue

`RecordOfValue` est mappé vers le type complexe suivant:

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

Choix de séquence d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.14 SetValue

`SetValue` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.

- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.15 SetOfValue

`SetOfValue` est mappé vers le type complexe suivant:

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

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

Choix de séquence d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.16 EnumeratedValue

`EnumeratedValue` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `element` La valeur d'énumération.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.17 UnionValue

`UnionValue` est mappé vers le type complexe suivant:

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

Choix d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.18 AnytypeValue

`AnytypeValue` est mappé vers le type complexe suivant:

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

Choix d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `address` Valeur d'adresse.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.3.19 AddressValue

`AddressValue` est mappé vers le type complexe suivant:

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

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

Choix d'éléments:

- `integer` Valeur d'entier.
- `float` Valeur en virgule flottante.
- `boolean` Valeur booléenne.
- `objid` Valeur d'identificateur d'objet.
- `verdicttype` Valeur de type de verdict.
- `bitstring` Valeur en chaîne binaire.
- `hexstring` Valeur en chaîne hexadécimale.
- `octetstring` Valeur en chaîne d'octets.
- `charstring` Valeur en chaîne de caractères.
- `universal_charstring` Valeur en chaîne de caractères universels.
- `record` Valeur d'enregistrement.
- `record_of` Valeur d'enregistrement de.
- `set` Valeur d'ensemble.
- `set_of` Valeur d'ensemble de.
- `enumerated` Valeur d'énumération.
- `union` Valeur de réunion.
- `anytype` Valeur de type quelconque.

Attributs:

- Les mêmes attributs que ceux de Value.

10.3.4 Mappage de types de journalisation abstraite

Des types additionnels sont définis afin de faciliter la journalisation de concordances entre valeurs et modèles.

10.3.4.1 TciValueTemplate

`TciValueTemplate` est mappé vers le type complexe suivant:

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

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

Choix d'éléments:

- `integer` Modèle en entier.
- `float` Modèle en virgule flottante.
- `boolean` Modèle booléen.
- `objid` Modèle d'identificateur d'objet.
- `verdicttype` Modèle de type de verdict.
- `bitstring` Modèle de chaîne binaire.
- `hexstring` Modèle de chaîne hexadécimale.
- `octetstring` Modèle de chaîne d'octets.
- `charstring` Modèle de chaîne de caractères.
- `universal_charstring` Modèle de chaîne de caractères universels.
- `record` Modèle d'enregistrement.
- `record_of` Modèle d'enregistrement de.
- `set` Modèle d'ensemble.
- `set_of` Modèle d'ensemble de.
- `enumerated` Modèle d'énumération.
- `union` Modèle de réunion.
- `anytype` Modèle de type quelconque.
- `adresse` Modèle d'adresse.
- `omit` Modèle d'omission.
- `any` Modèle d'objet quelconque.
- `anyoromit` Modèle d'objet quelconque ou d'omission.
- `templateDef` Définition de modèle complexe.

Attributs:

- néant.

10.3.4.2 TciNonValueTemplate

`TciNonValueTemplate` est mappé vers le type complexe suivant:

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

Choix d'éléments:

- `any` Modèle d'objet quelconque.
- `all` Modèle de totalité d'objets.
- `templateDef` Définition de modèle complexe.

Attributs:

- néant.

10.3.4.3 TciValueList

`TciValueList` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `val` Les valeurs contenues dans la liste des valeurs.

Attributs:

- néant.

10.3.4.4 TciValueDifference

`TciValueDifference` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `desc` Raison de la discordance.
- `val` Référence à la valeur discordante.
- `tmpl` Référence au modèle.

Attributs:

- néant.

10.3.4.5 TciValueDifferenceList

`TciValueDifferenceList` est mappé vers le type complexe suivant:

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

Séquence d'éléments:

- `diff` Différences de valeur/modèle contenues dans la liste des différences de valeur.

Attributs:

- néant.

10.4 Mappage des opérations à l'interface de journalisation

Chaque opération fournie à l'interface de journalisation possède une définition correspondante de type complexe dans le langage XML. Ces définitions de type complexe sont des extensions du type "Event" (événement).

10.4.1 Event

`Event` est mappé vers le type complexe suivant:

```
<!-- définition commune pour tous les événements -->
<xsd:complexType name="Event" mixed="true">
    <xsd:sequence>
        <xsd:element name="am" type="SimpleTypes:TString"/>
    </xsd:sequence>
    <xsd:attribute name="ts" type="xsd:time" use="required"/>
    <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
    <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

<!-- structure générale des identificateurs pour composants de test, ports et temporisateurs -->
    <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
    <xsd:attribute name="id" type="SimpleTypes:TInteger" use="required"/>
    <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
</xsd:complexType>
```

Eléments:

- `am` Message à utiliser pour information complémentaire dans le journal.

Attributs:

- `ts` Instant auquel l'événement est produit.
- `src` Fichier source de la spécification de test.
- `line` Numéro de la ligne où la requête est effectuée.
- `name` Nom du composant qui produit cet événement.
- `id` Identificateur du composant qui produit cet événement.
- `type` Type du composant qui produit cet événement.

10.4.2 Mappage d'opérations

Le mappage des opérations est indiqué dans ce qui suit.

Sous-interface TCI-TL fournie	

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

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

void tliTcStop(in TString am, in TInteger ts,
in TString src, in TInteger line,
in TriComponentIdType c)
	`<xsd:complexType name="tliTcStop">`
	`<xsd:complexContent mixed="true">`
	`<xsd:extension base="Events:Event"/>`
	`</xsd:complexContent>`
	`</xsd:complexType>`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

11 Scénarios d'utilisation

Le présent paragraphe contient des scénarios d'utilisation qui devraient aider les utilisateurs de l'interface TCI, ainsi que les vendeurs d'utilitaires qui fournissent l'interface TCI, à comprendre la sémantique des opérations définies dans la présente Recommandation.

Les scénarios sont définis en termes de diagrammes séquentiels en langage UML, qui montrent les interactions entre les entités d'interface TCI. Les scénarios sont expliqués et, le cas échéant, sont complétés par un fragment en notation TTCN-3 correspondant au scénario.

11.1 Initialisation, collecte des informations, journalisation

11.1.1 Scénario d'utilisation: initialisation

Le scénario décrit dans la Figure 9 montre la phase d'initialisation pour un système de test quand un module TTCN-3 doit être sélectionné pour exécution. Tout d'abord, un module radical doit être posé avec le paramètre `tciRootModule`. Les paramètres du module radical peuvent être obtenus avec le paramètre `tciGetModuleParameters`. Les informations relatives aux paramètres de module peuvent servir à demander à l'utilisateur du système de test des valeurs concrètes pour chaque paramètre de module. La liste de tests élémentaires disponibles dans le module radical peut être extraite par l'opération `tciGetTestCases`. Ces tests élémentaires peuvent être directement exécutés à partir de la gestion de test. Leurs paramètres et leur interface avec le système de test peuvent respectivement être obtenus par les opérations `tciGetTestCaseParameters` et `tciGetTestCaseTSI`.

11.1.1.1 Diagramme séquentiel

Figure 9/Z.145 – Scénario d'utilisation – Initialisation

11.1.1.2 Fragment TTCN-3

L'initialisation est hors du domaine d'application de la notation TTCN-3.

11.1.2 Scénario d'utilisation: demande des paramètres de module

Le scénario décrit dans la Figure 10 montre comment un composant de test demande la valeur réelle d'un paramètre de module requis pour l'exécution de son comportement de test. Tout d'abord, le type d'un paramètre de module est demandé, puis la valeur peut être construite par la gestion TM et communiquée à l'exécutable TE.

11.1.2.1 Diagramme séquentiel

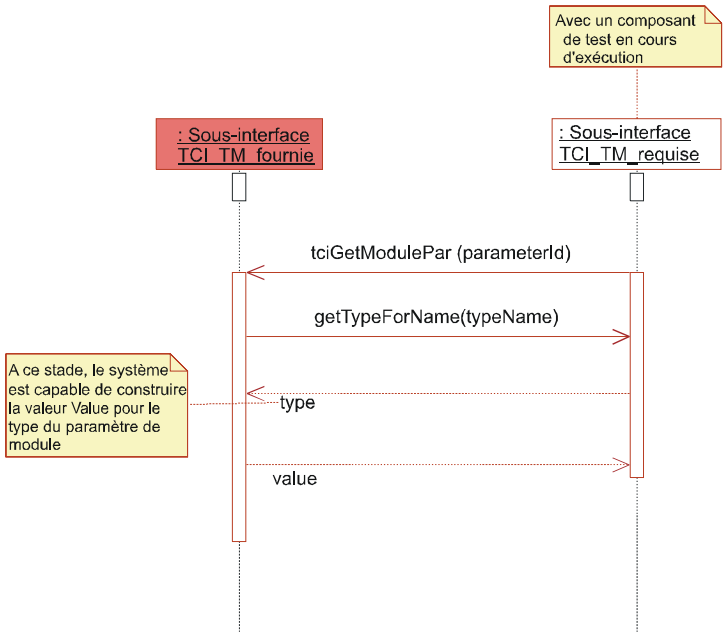

Figure 10/Z.145 – Scénario d'utilisation – Requête des paramètres du module

11.1.2.2 Fragment TTCN-3

```
module AModule {
    …
    modulepar {
        integer AModulePar
    }
    …
    function AFunction (…) … {
        integer x;
        …
        x:= 2+AModulePar; // une expression avec un paramètre de module
        …
    }
    …
}
```

11.1.3 Scénario d'utilisation: journalisation

Le scénario décrit dans la Figure 11 montre la journalisation des informations pendant l'exécution d'un comportement de test par un composant de test. Le message à journaliser est propagé vers la journalisation de test.

11.1.3.1 Diagramme séquentiel

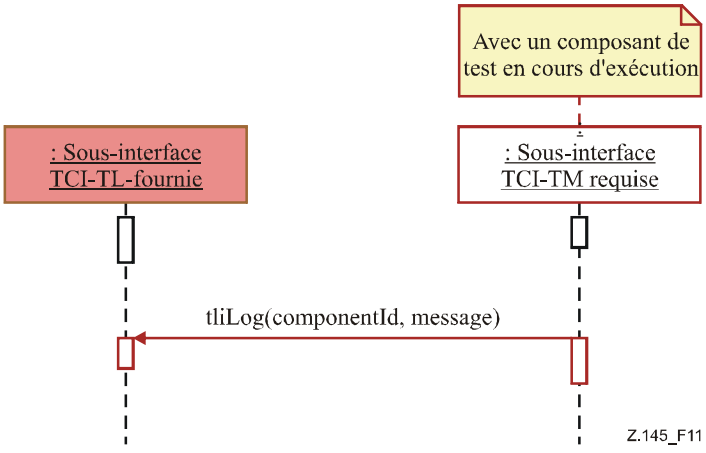

Figure 11/Z.145 – Scénario d'utilisation – Journalisation

11.1.3.2 Fragment TTCN-3

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

11.2 Exécution de test élémentaire et commande

11.2.1 Scénario d'utilisation: exécution de commande

Le scénario décrit dans la Figure 12 montre la séquence d'opérations permettant d'exécuter la partie commande d'un module TTCN-3. Le module contenant la partie commande est d'abord sélectionné, puis la commande est lancée, puis il est exécuté jusqu'à ce que son exécution soit terminée par l'exécutable TE.

11.2.1.1 Diagramme séquentiel

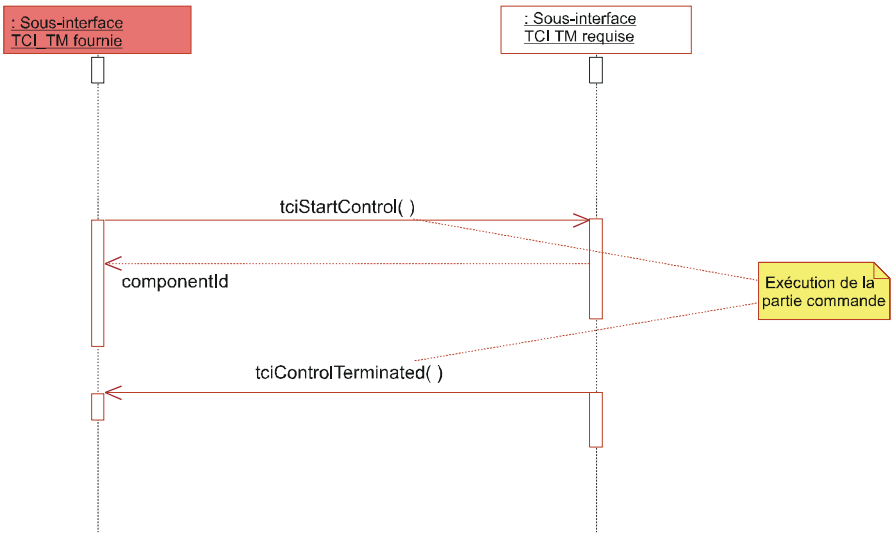

Figure 12/Z.145 – Scénario d'utilisation – Exécution de commande

11.2.1.2 Fragment TTCN-3

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

11.2.2 Scénario d'utilisation: exécution de test élémentaire dans le cadre d'une commande

Le scénario décrit dans la Figure 13 montre comment un test élémentaire est exécuté dans la partie commande.

11.2.2.1 Diagramme séquentiel

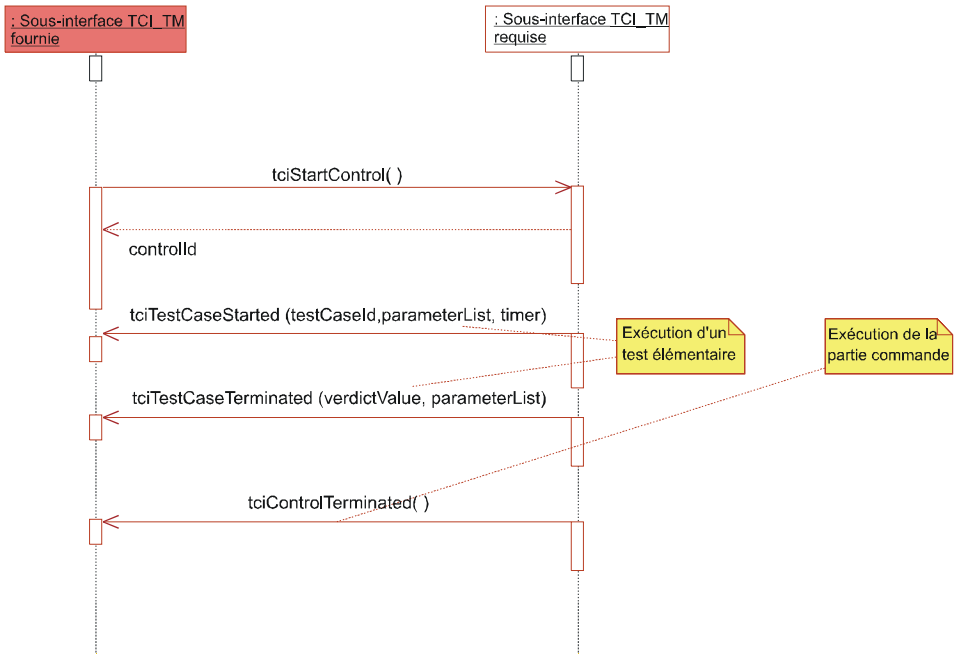

Figure 13/Z.145 – Scénario d'utilisation – Exécution de test élémentaire dans le cadre d'une commande

11.2.2.2 Fragment TTCN-3

```
module AModule {
    …
    testcase ATestCase(…)… {
        … // le comportement du test élémentaire
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.2.3 Scénario d'utilisation: exécution directe de test élémentaire

Le scénario décrit dans la Figure 14 montre comment un test élémentaire peut être directement exécuté à partir de la gestion de test à l'extérieur la partie commande. Après sélection du module TTCN-3 contenant le test élémentaire à exécuter, le lancement de test élémentaire est demandé. Quand le test élémentaire termine son exécution, la gestion de test est informée par l'exécutable TE de la terminaison du test élémentaire.

11.2.3.1 Diagramme séquentiel

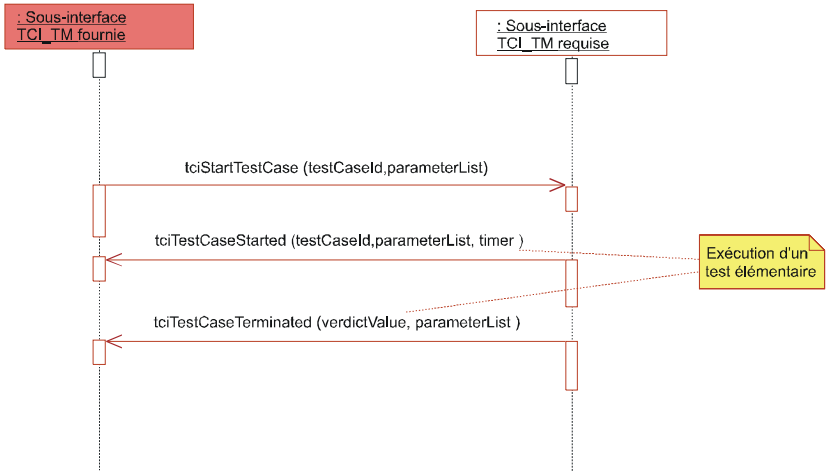

Figure 14/Z.145 –Scénario d'utilisation – Exécution directe de test élémentaire

11.2.3.2 Fragment TTCN-3

L'exécution directe d'un test élémentaire est hors du domaine d'application de la notation TTCN-3.

11.2.4 Scénario d'utilisation: exécution de test élémentaire par interface TRI

Le scénario décrit dans la Figure 15 montre comment l'interface TRI est informée de l'exécution d'un test élémentaire de façon qu'elle puisse établir et initialiser les ports du système en temps voulu. La requête de test élémentaire doit être propagée avant le lancement du comportement de test sur le composant MTC du test élémentaire en cours.

11.2.4.1 Diagramme séquentiel

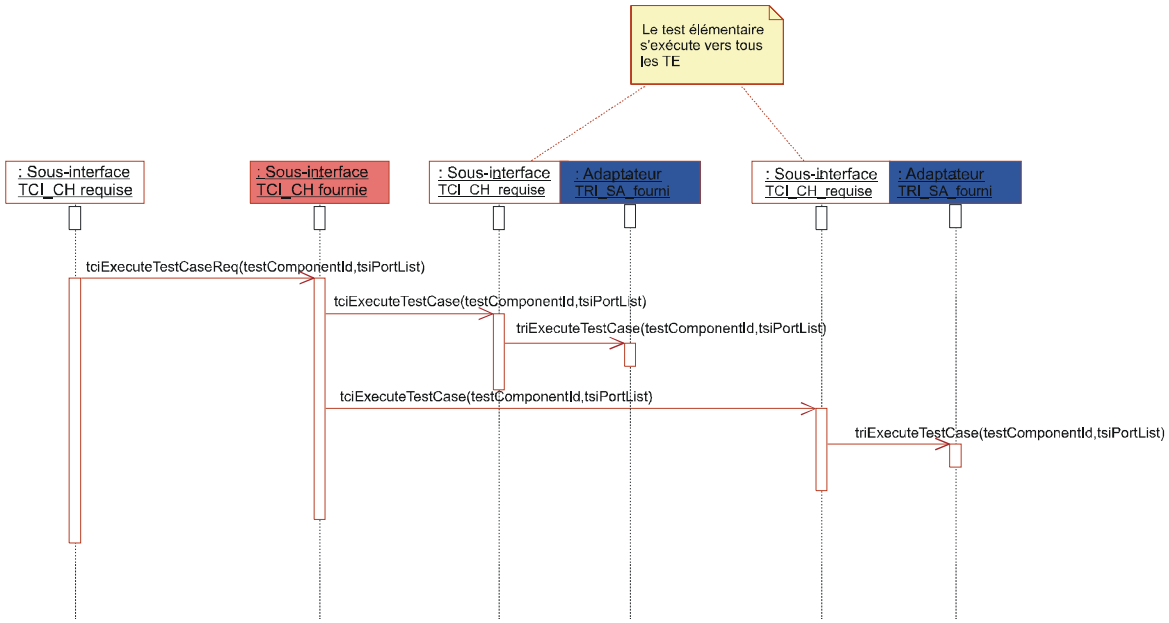

Figure 15/Z.145 –Scénario d'utilisation – Exécution de test élémentaire par interface TRI

11.2.4.2 Fragment TTCN-3

```
module AModule {
    …
    testcase ATestCase(…)… {
        … // le comportement du test élémentaire
    }
    …
    control {
        …
```

```
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3 Traitement de composant

11.3.1 Scénario d'utilisation: création d'un composant de commande locale

Le scénario décrit dans la Figure 16 démontre la création du composant de commande dans le nœud où réside l'interface avec l'utilisateur de la gestion de test TCI-TM. Un composant de commande est créé chaque fois que la partie commande d'un module TTCN-3 est exécutée. Chaque fois que la gestion de test TCI-TM effectue le lancement de la partie commande, une requête de création de composant de test est envoyée à la sous-interface TCI-CH, qui la propage vers l'exécutable TE où le composant de commande devrait être créé. Dans ce cas, il s'agit de l'exécutable TE situé dans le même nœud. L'identificateur du composant de commande est renvoyé et indiqué à la sous-interface TCI-TM. L'identificateur est alors utilisé afin de lancer le comportement de la partie commande sur le composant de commande.

11.3.1.1 Diagramme séquentiel

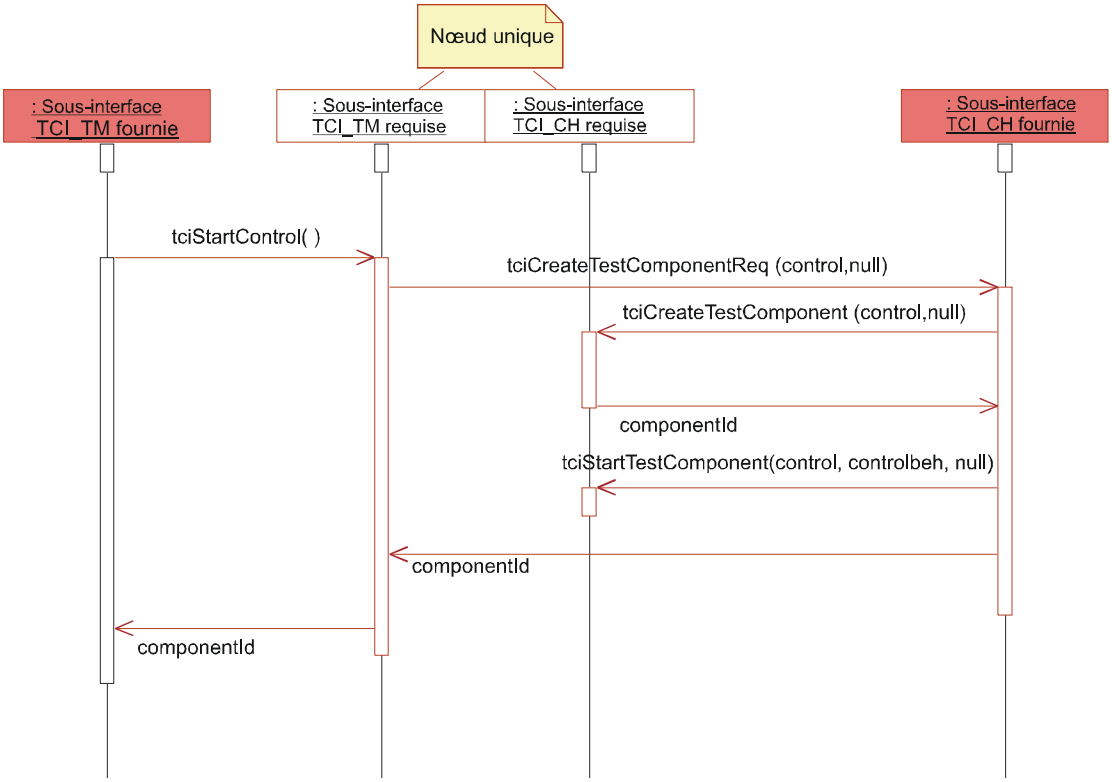

Figure 16/Z.145 – Scénario d'utilisation – Création d'un composant de commande locale

11.3.1.2 Fragment TTCN-3

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

11.3.2 Scénario d'utilisation: création d'un composant de télécommande

Le scénario décrit dans la Figure 17 démontre la création du composant de commande dans un autre nœud que celui où réside l'interface avec l'utilisateur de la gestion de test TCI-TM. Un composant de commande est créé chaque fois que la partie commande d'un module TTCN-3 est exécutée. Chaque fois que la gestion de test TCI-TM effectue le lancement de la partie commande, une requête de création de composant de test est envoyée à la sous-interface TCI-CH, qui la

propage vers l'exécutable TE où le composant de commande devrait être créé. Dans ce cas, il s'agit de l'exécutable TE situé dans un autre nœud distant. L'identificateur du composant de commande est renvoyé et indiqué à la sous-interface TCI-TM. Cet identificateur est alors utilisé afin de lancer le comportement de la partie commande sur le composant de commande.

11.3.2.1 Diagramme séquentiel

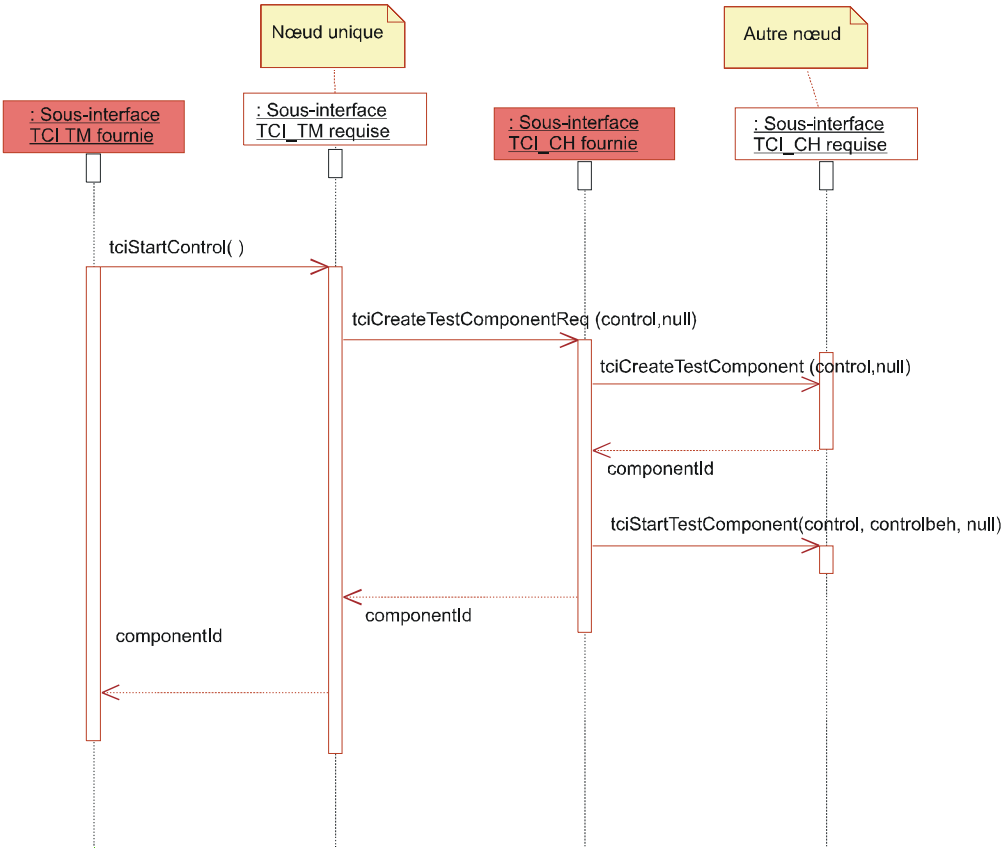

Figure 17/Z.145 – Scénario d'utilisation – Création d'un composant de télécommande

11.3.2.2 Fragment TTCN-3

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

11.3.3 Scénario d'utilisation: création d'un composant MTC local

Le scénario décrit dans la Figure 18 démontre la création locale du composant de test principal. Le terme "locale" vise deux cas:

1) dans le même nœud que celui où réside l'interface avec l'utilisateur de la gestion de test TCI-TM (quand un test élémentaire est lancé directement);

2) dans le même nœud que celui où réside le composant de commande (quand un test élémentaire est exécuté à partir d'une partie commande).

Un composant de test principal est créé chaque fois qu'un test élémentaire est exécuté: une requête de création de composant de test est envoyée à la sous-interface TCI-CH, qui la propage vers l'exécutable TE où le composant de test principal devrait être créé. Dans ce cas, il s'agit de l'exécutable TE situé dans le même nœud. L'identificateur du composant de test principal est renvoyé et indiqué à la sous-interface TCI-TM. Cet identificateur est alors utilisé afin de lancer le comportement de test élémentaire sur le composant de test principal (ce cas de figure n'est pas représenté ici mais est traité de la même façon que dans les scénarios décrits dans les § 11.3.5 et 11.3.6).

11.3.3.1 Diagramme séquentiel

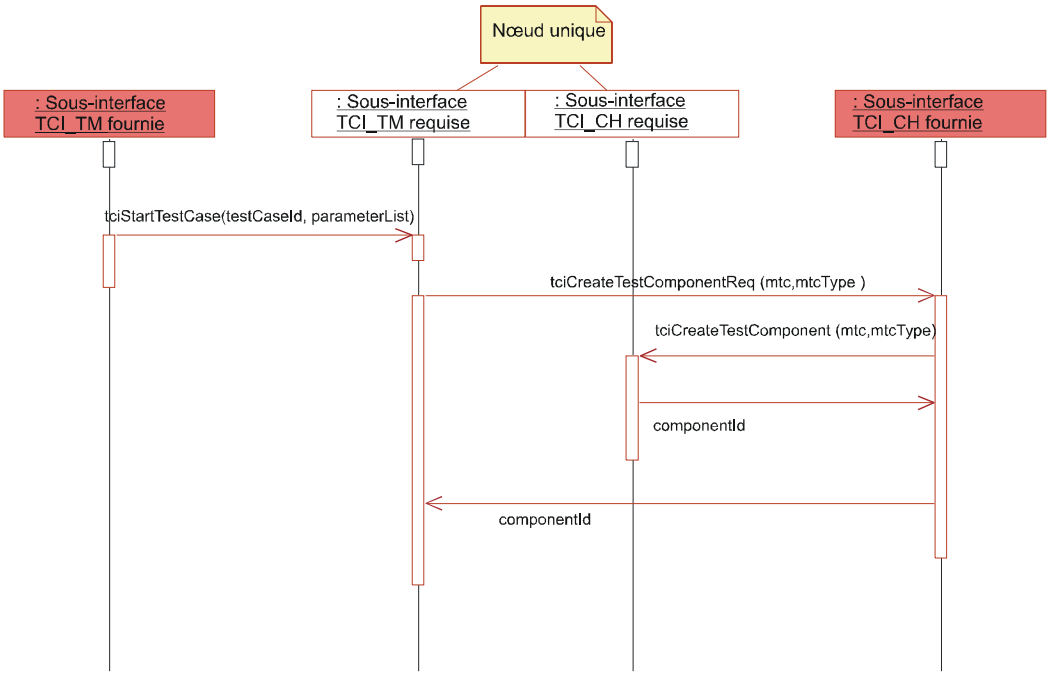

Figure 18/Z.145 – Scénario d'utilisation – Création d'un composant MTC local

11.3.3.2 Fragment TTCN-3

```
module AModule {
    …
    testcase ATestCase(…)runs on MTCType … {
        … // le comportement du test élémentaire
    }
    …
}
```

11.3.4 Scénario d'utilisation: création à distance d'un composant MTC

Le scénario décrit dans la Figure 19 démontre la création à distance du composant de test principal. Le terme "à distance" vise deux cas:

1) dans un autre nœud que celui où réside l'interface avec l'utilisateur de la gestion de test TCI-TM (quand un test élémentaire est lancé directement);
2) dans un autre nœud que celui où réside le composant de commande (quand un test élémentaire est exécuté à partir d'une partie commande).

Un composant de test principal est créé chaque fois qu'un test élémentaire est exécuté: une requête de création de composant de test est envoyée à la sous-interface TCI-CH, qui la propage vers l'exécutable TE où le composant de test principal devrait être créé. Dans ce cas, il s'agit l'exécutable TE situé dans un autre nœud. L'identificateur du composant de test principal est renvoyé et indiqué à la sous-interface TCI-TM. Cet identificateur est alors utilisé afin de lancer le comportement de test élémentaire sur le composant de test principal (ce cas de figure n'est pas représenté ici mais est traité de la même façon que dans les scénarios décrits dans les § 11.3.5 et 11.3.6).

11.3.4.1 Diagramme séquentiel

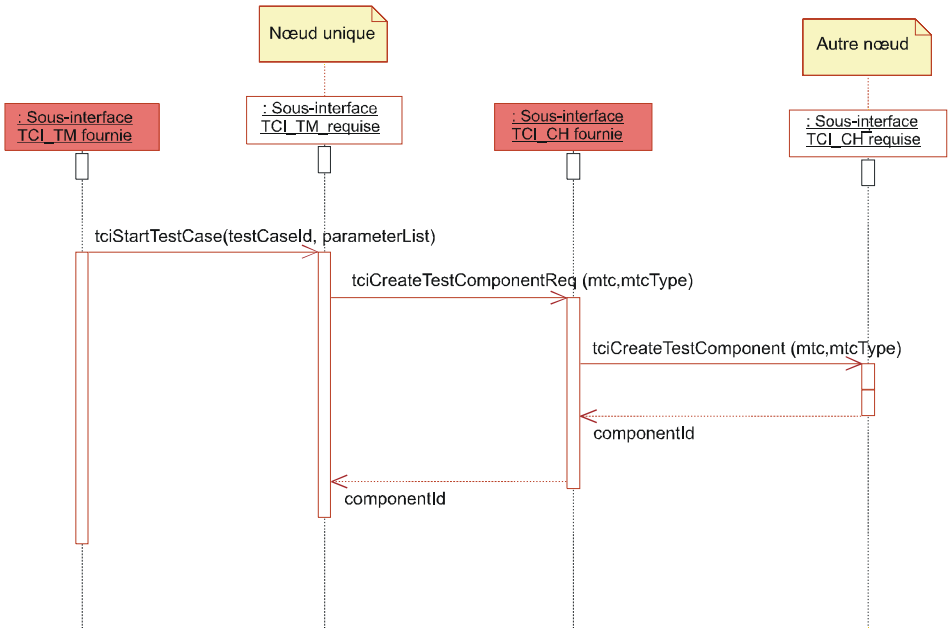

Figure 19/Z.145 – Scénario d'utilisation – Création à distance d'un composant MTC

11.3.4.2 Fragment TTCN-3

```
module AModule {
    …
    testcase ATestCase(…)runs on MTCType … {
        … // le comportement du test élémentaire
    }
    …
}
```

11.3.5 Scénario d'utilisation: traitement de composant pour exécution de test élémentaire dans le cadre d'une commande

Le scénario décrit dans la Figure 20 démontre le traitement de composants pour l'exécution de test élémentaire d'une partie commande. Quand la partie commande est lancée, un composant de commande est créé et son identificateur de composant est renvoyé à la gestion de test. Pour chaque test élémentaire à exécuter dans la partie commande, un composant de test principal est créé et l'identificateur de composant est renvoyé au composant de commande. Ensuite, le comportement de test élémentaire est lancé dans le composant de test principal et la gestion de test est informée du lancement de test élémentaire. Quand le composant de test principal arrive en fin d'exécution, une requête de terminaison du composant de test principal est propagée de concert avec le verdict local du composant de test principal afin de permettre le calcul du verdict de test global et afin de transmettre les informations sur la terminaison du test élémentaire.

11.3.5.1 Diagramme séquentiel

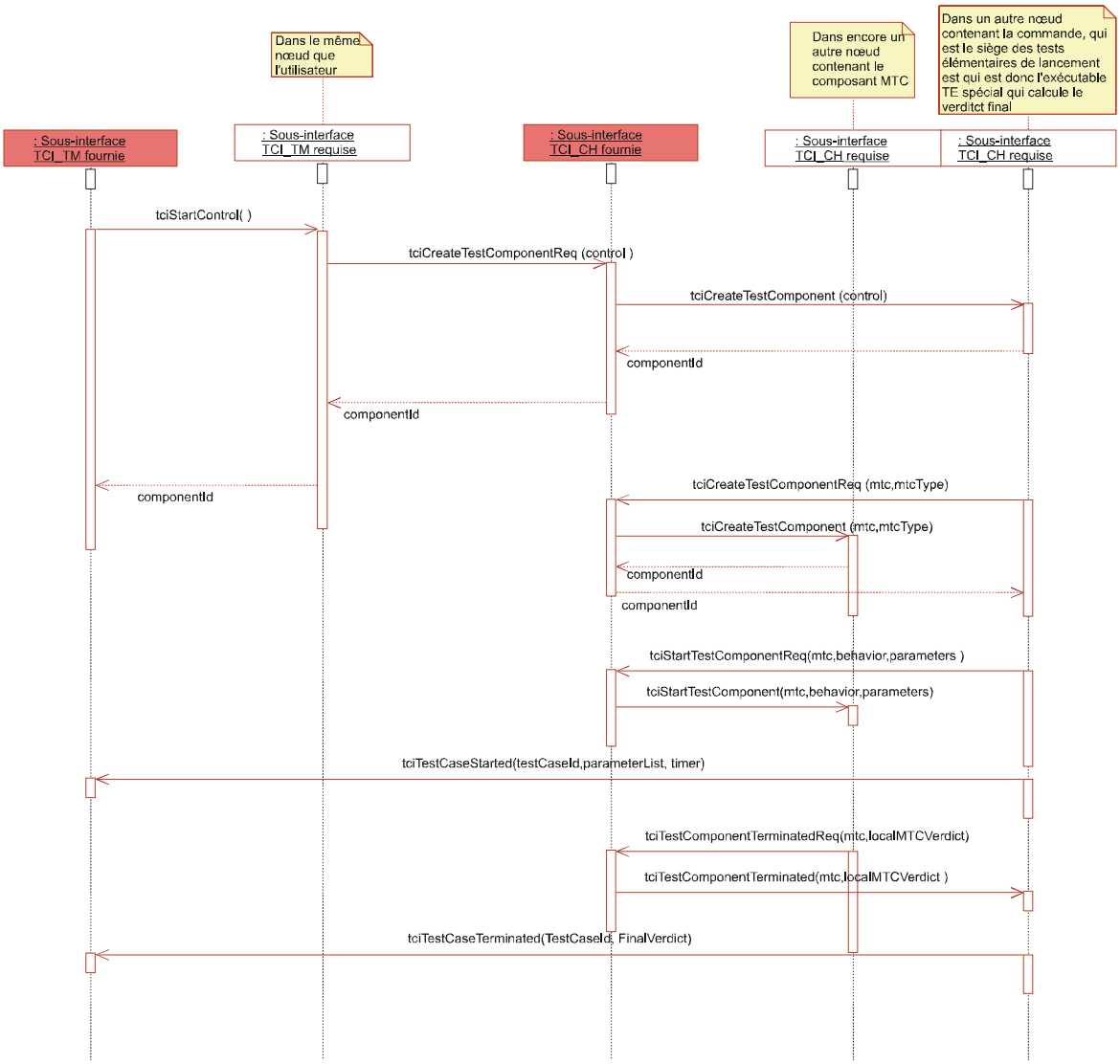

Figure 20/Z.145 – Scénario d'utilisation: traitement de composant pour exécution de test élémentaire dans le cadre d'une commande

11.3.5.2 Fragment TTCN-3

```
module AModule {
    …
    testcase ATestCase(…)… {
        … //le comportement du test élémentaire
    }
    …
    control {
        …
        execute(ATestCase(…));
        …
    }
    …
}
```

11.3.6 Scénario d'utilisation: traitement de composant pour exécution directe de test élémentaire

Le scénario décrit dans la Figure 21 montre comment des composants de test sont traités quand un test élémentaire est exécuté directement, c'est-à-dire à l'extérieur d'une partie commande. Quand un test élémentaire est lancé, le composant de test principal est créé et le comportement de test élémentaire est d'abord lancé dans ce composant de test principal. Chaque fois qu'un composant de test parallèle est utilisé à l'intérieur d'un test élémentaire, il est traité de la même façon: le composant de test parallèle est lancé d'abord, ce qui envoie une requête de création de composant de test à l'entité de

sous-interface TCI-CH, laquelle propage l'opération de création de composant de test vers l'exécutable TE dans lequel le composant de test parallèle doit être créé. L'identificateur du composant de test parallèle qui a été créé est renvoyé. Cet identificateur est alors utilisé afin de lancer le comportement de composant PTC pour l'opération de lancement. Quand le composant PTC arrive en fin d'exécution, une requête de terminaison de composant de test est propagée en même temps que le verdict de test local afin d'informer la sous-interface TCI-CH de cette terminaison. La même procédure est suivie quand le composant de test principal arrive en fin d'exécution. En outre, la terminaison du composant de test principal se traduit par la terminaison globale du test élémentaire.

11.3.6.1 Diagramme séquentiel

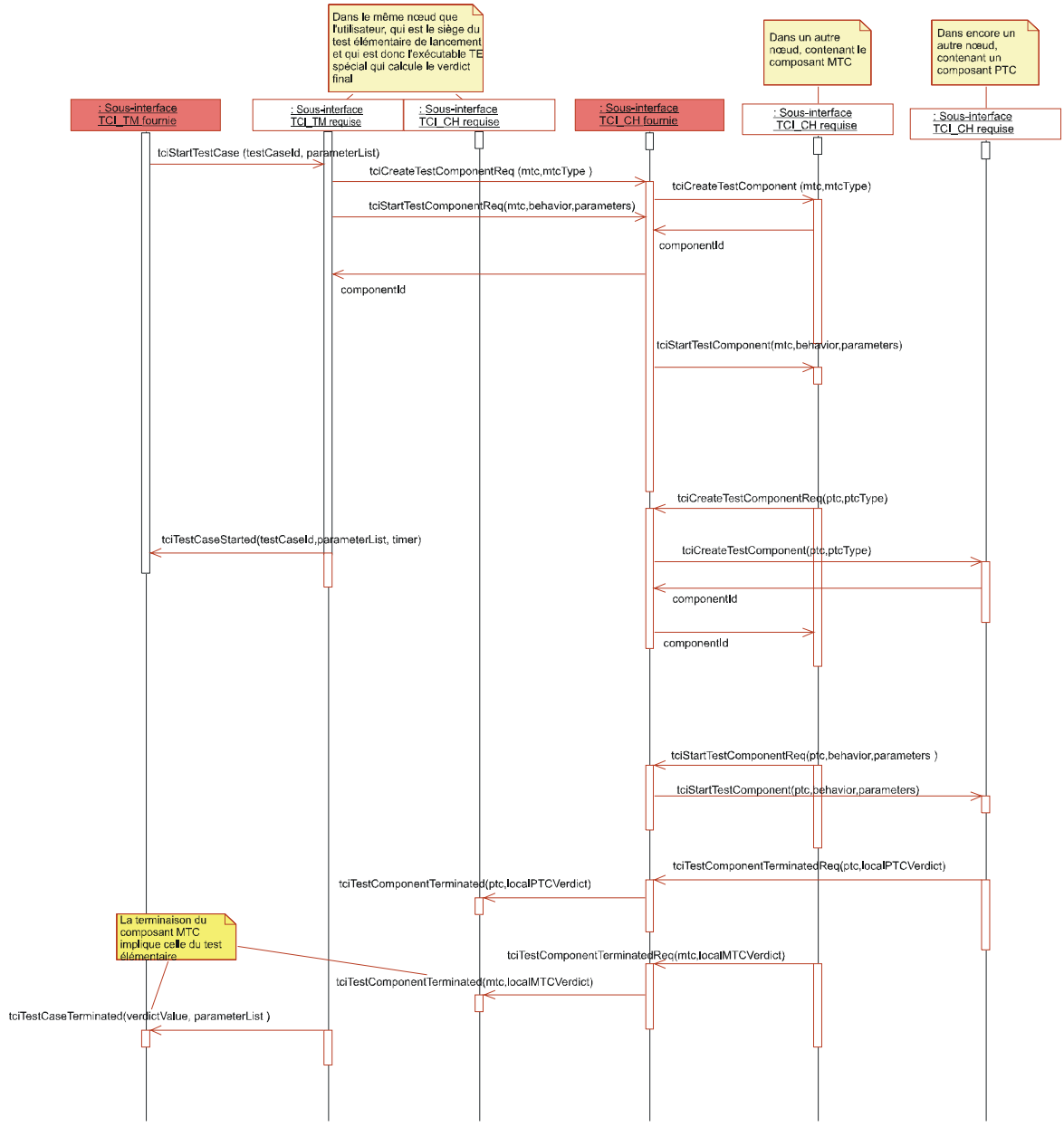

Figure 21/Z.145 – Scénario d'utilisation: traitement de composant pour exécution directe de test élémentaire

11.3.6.2 Fragment TTCN-3

```
module AModule {
    …
    function APTCBehaviour(…) runs on APTCType {
        … //le comportement du composant PTC
    }
    …
    testcase ATestCase(…)… {
        … //le comportement du test élémentaire
```

```
        var APTCType PTC:= APTCType.create;
        …
        PTC.start(APTCBehaviour(…));
        …
    }
    …
}
```

11.3.7 Scénario d'utilisation: propagation d'opération de mappage/connexion

Le scénario décrit dans la Figure 22 montre comment les ports sont mappés. La requête visant à mapper un port est propagée vers l'exécutable TE où le mappage est finalement effectué. La propagation des requêtes de connexion fonctionne de la même façon.

11.3.7.1 Diagramme séquentiel

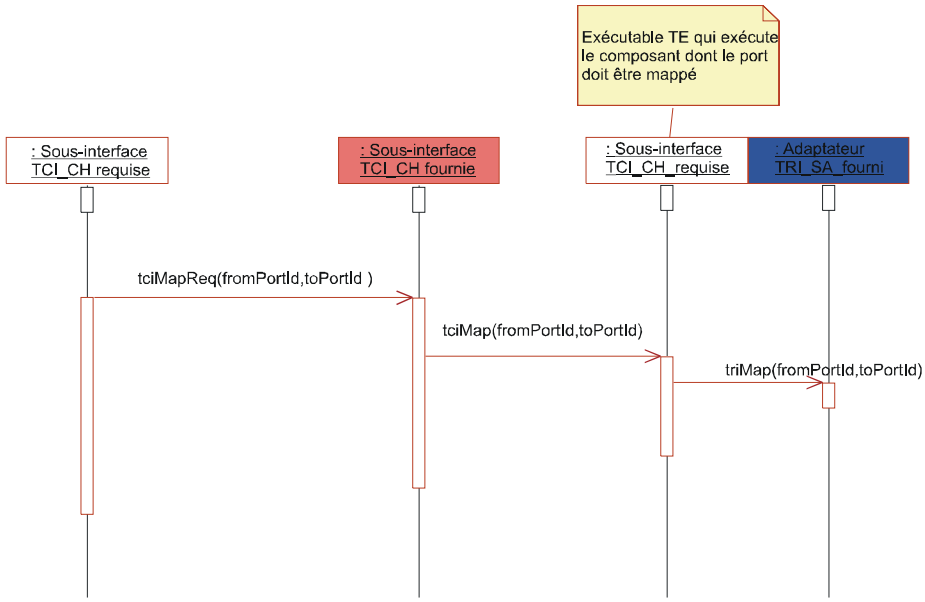

Figure 22/Z.145 – Scénario d'utilisation: propagation de mappage

11.3.7.2 Fragment TTCN-3

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

11.3.8 Scénario d'utilisation: propagation d'une opération de démappage/déconnexion

Le scénario décrit dans la Figure 23 montre comment les ports sont démappés. La requête visant à démapper un port est propagée vers l'exécutable TE où le démappage est finalement effectué. La propagation des requêtes de déconnexion fonctionne de la même façon.

11.3.8.1 Diagramme séquentiel

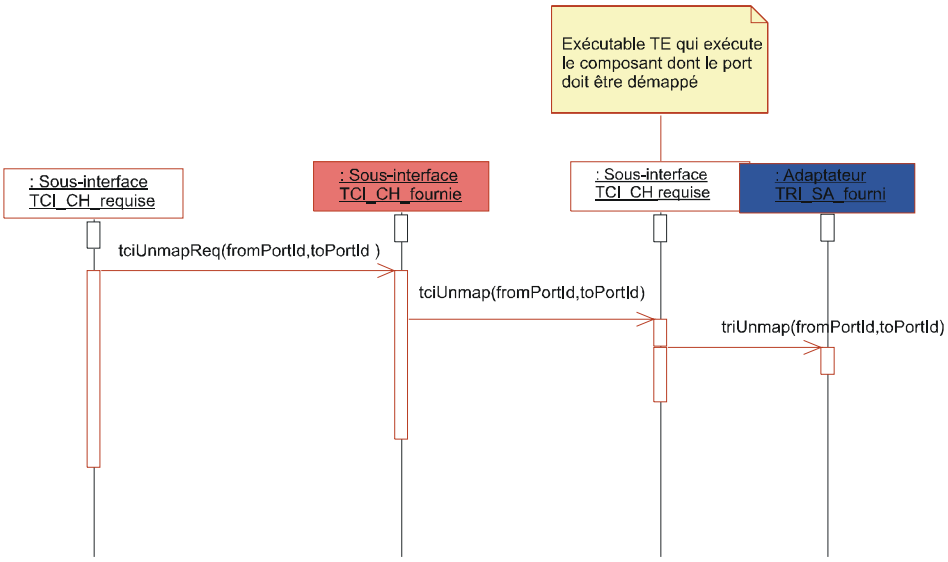

Figure 23/Z.145 – Scénario d'utilisation – Propagation de démappage

11.3.8.2 Fragment TTCN-3

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

11.4 Terminaison de test élémentaire et commande

11.4.1 Scénario d'utilisation: arrêt d'un test élémentaire

Le scénario décrit dans la Figure 24 montre comment un test élémentaire est arrêté à partir de la gestion de test pendant l'exécution d'un test élémentaire. Une fois que la gestion TM a reçu des informations sur un test élémentaire qui a été lancé, un arrêt de test élémentaire peut être demandé jusqu'à réception de la notification de terminaison du test élémentaire. Lors de l'arrêt d'un test élémentaire, tous les composants de test parallèles seront arrêtés et le système de test sera réinitialisé.

11.4.1.1 Diagramme séquentiel

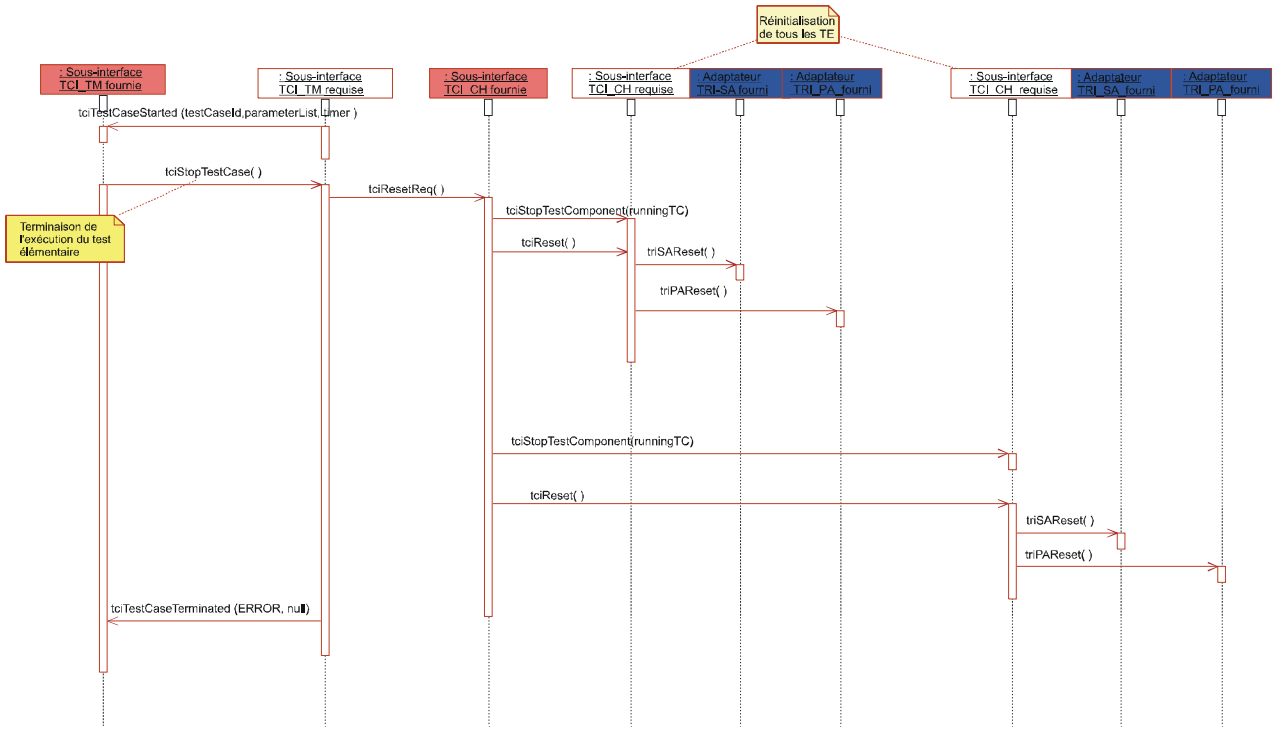

Figure 24/Z.145 – Scénario d'utilisation – Arrêt d'un test élémentaire

11.4.1.2 Fragment TTCN-3

Il n'y a aucun code TTCN-3 associé à la façon dont la gestion TM choisit d'implémenter la terminaison du test élémentaire. Ce choix est hors du domaine d'application de la notation TTCN-3.

11.4.2 Scénario d'utilisation: arrêt d'une commande

Le scénario décrit dans la Figure 25 montre comment une partie commande est arrêtée à partir de la gestion de test pendant l'exécution de la partie commande. Une partie commande peut être arrêtée entre le lancement de la commande et sa terminaison. Si la partie commande reçoit une requête d'arrêt de test élémentaire alors qu'un test élémentaire est en cours d'exécution, le test élémentaire en cours d'exécution doit être arrêté. Par ailleurs, le système de test doit être réinitialisé comme décrit dans la Figure 24.

11.4.2.1 Diagramme séquentiel

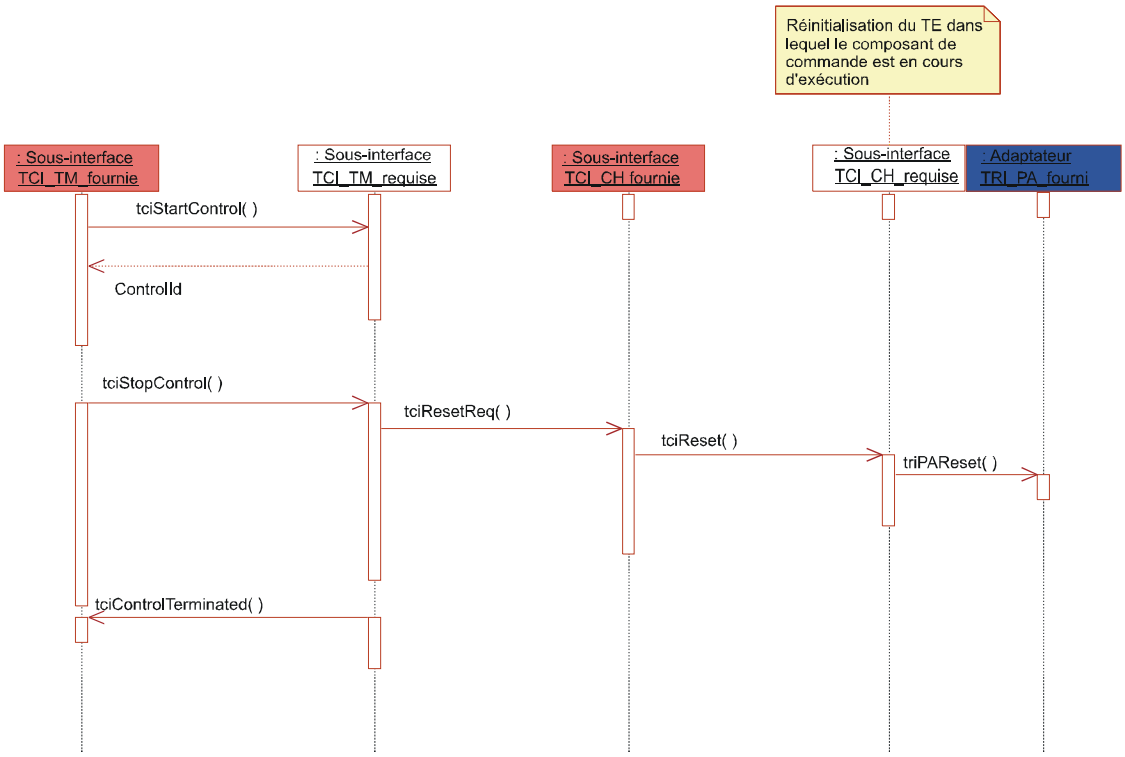

Figure 25/Z.145 – Scénario d'utilisation – Arrêt d'une commande

11.4.2.2 Fragment TTCN-3

L'opération d'arrêt d'une partie commande à partir de la gestion de test est hors du domaine d'application de la notation TTCN-3 de telle sorte qu'aucun fragment TTCN-3 n'existe.

11.4.3 Scénario d'utilisation: terminaison de commande après erreur

Le scénario décrit dans la Figure 26 montre le traitement de situations d'erreur pendant l'exécution d'une partie commande quand aucun test élémentaire n'est en cours d'exécution. La gestion de test est informée de la situation d'erreur et doit alors mettre fin à l'exécution de la partie commande explicitement. Le système de test sera réinitialisé dès la terminaison de la partie commande.

11.4.3.1 Diagramme séquentiel

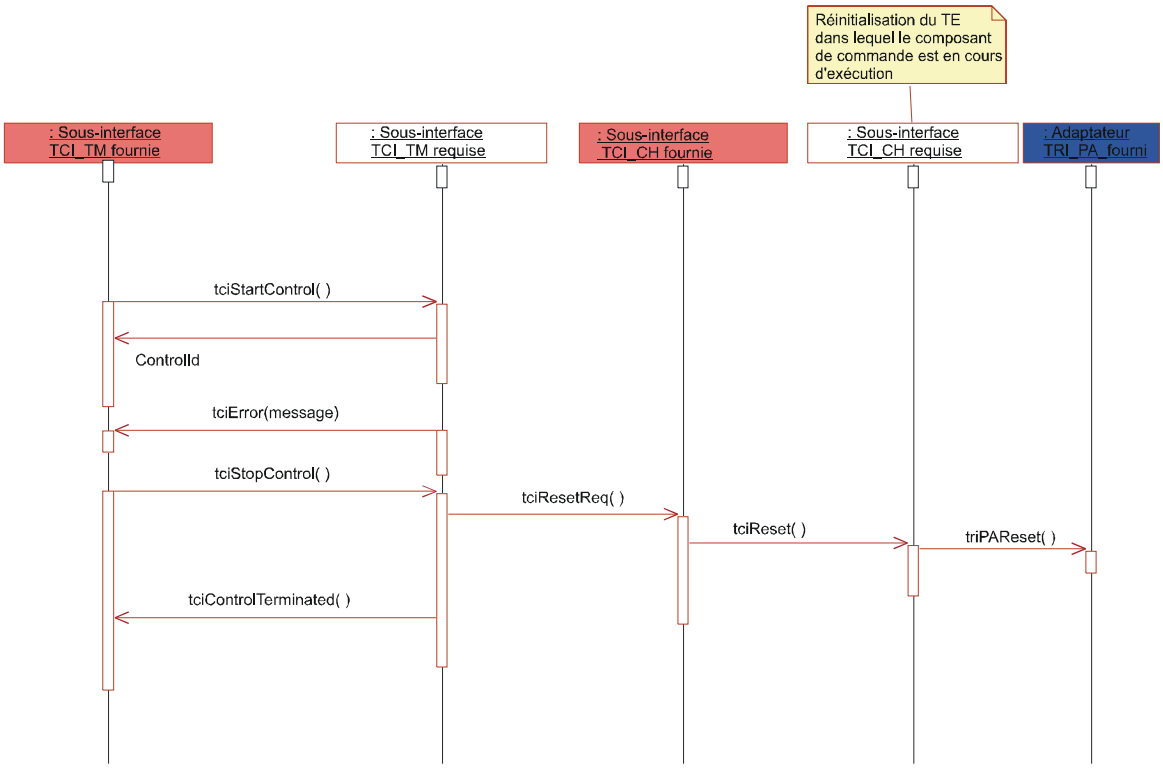

Figure 26/Z.145 – Scénario d'utilisation – Terminaison de commande après erreur

11.4.3.2 Fragment TTCN-3

Il n'y a aucun fragment TTCN-3 pour ce scénario, étant donné que les situations d'erreur sont des cas exceptionnels dans un système de test et non un concept de notation TTCN-3 en soi. En revanche, la sémantique de la notation TTCN-3 décrit diverses situations possibles d'erreur dans un système de test.

11.4.4 Scénario d'utilisation: terminaison d'un test élémentaire après erreur

Le scénario décrit dans la Figure 27 montre le traitement de situations d'erreur pendant l'exécution directe d'un test élémentaire. La gestion de test est informée de la situation d'erreur. La gestion TM doit alors explicitement terminer l'exécution du test élémentaire. Lors de l'arrêt d'un test élémentaire, les composants de test parallèles seront arrêtés et le système de test sera réinitialisé.

11.4.4.1 Diagramme séquentiel

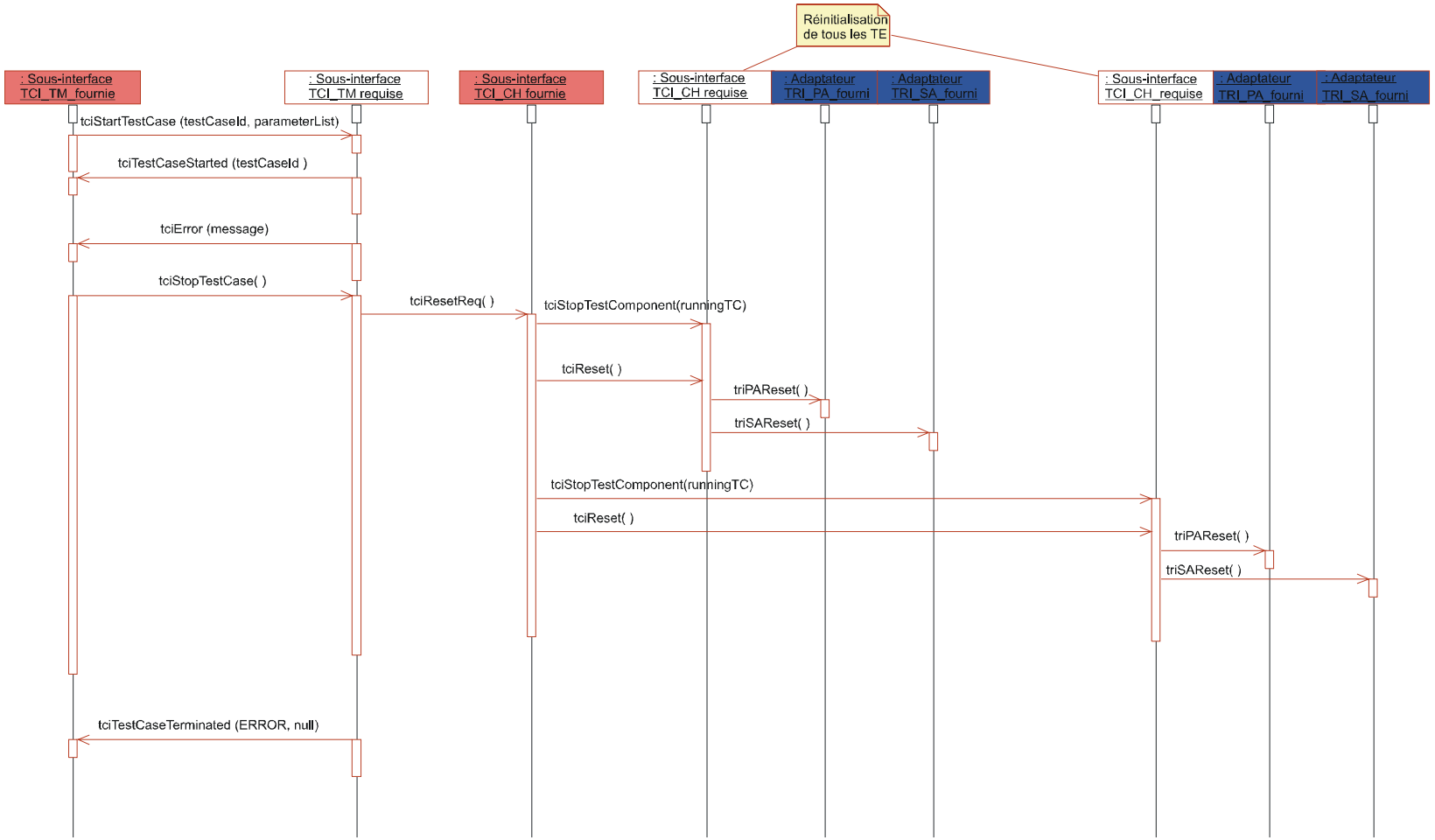

Figure 27/Z.145 – Scénario d'utilisation – Terminaison d'un test élémentaire après erreur

11.4.4.2 Fragment TTCN-3

Il n'y a aucun fragment TTCN-3 pour ce scénario, étant donné que les situations d'erreur sont des cas exceptionnels d'un système de test et non un concept de notation TTCN-3 en soi. En revanche, la sémantique de la notation TTCN-3 décrit diverses situations possibles d'erreur dans un système de test.

11.4.5 Scénario d'utilisation: réinitialisation

Le scénario décrit dans la Figure 28 montre la réinitialisation du système de test. Dans ce cas, tous les exécutables TE, ainsi que leurs adaptateurs de système (SA, *system adaptor*) à l'interface TRI et leurs adaptateurs de plate-forme (PA, *platform adaptor*) à l'interface TRI sont réinitialisés.

11.4.5.1 Diagramme séquentiel

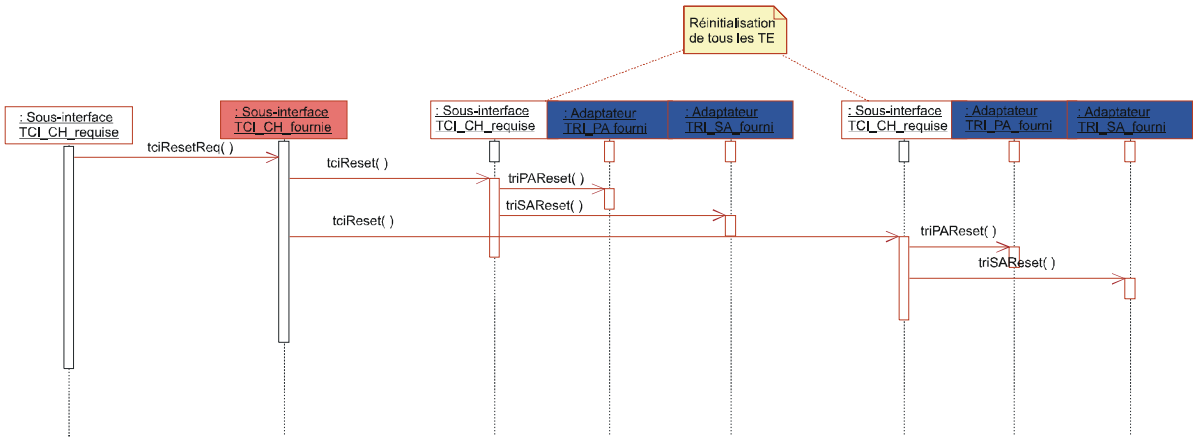

Figure 28/Z.145 – Scénario d'utilisation – Réinitialisation

11.4.5.2 Fragment TTCN-3

Il n'y a aucun fragment TTCN-3 pour ce scénario, étant donné que la réinitialisation requise après des situations d'erreur sont des cas exceptionnels dans un système de test et non un concept de notation TTCN-3 en soi.

11.5 Communication

11.5.1 Scénario d'utilisation: communication locale entre composants

Le scénario décrit dans la Figure 29 montre la communication entre composants de test (composant de test principal ou composants de test parallèles) qui résident dans le même nœud. Une requête de communication est envoyée à la sous-interface TCI-CH, qui décide alors de l'endroit où il convient de mettre en file d'attente ce modèle de communication. Dans ce cas, la communication est effectuée localement via l'exécutable TE, dans le même nœud. Le scénario montre une communication composée de messages, utilisant l'opération d'expédition: ce scénario est le même que pour les opérations d'appel, de réponse et de propagation d'exception.

11.5.1.1 Diagramme séquentiel

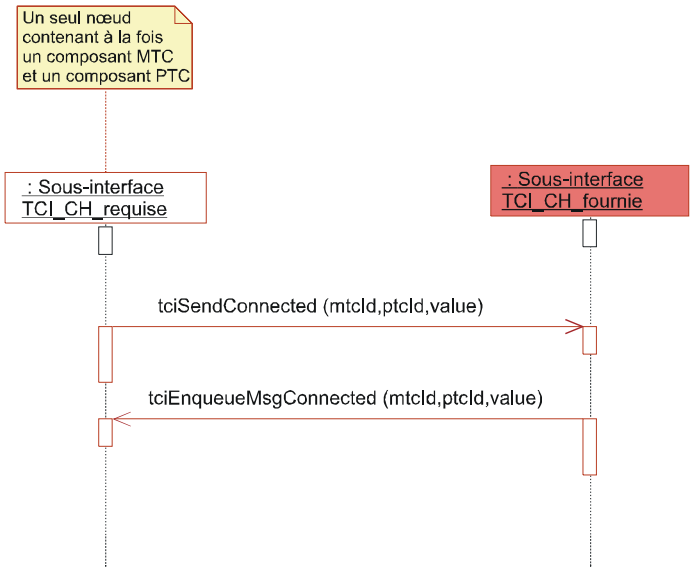

Figure 29/Z.145 – Scénario d'utilisation – Communication locale entre composants

11.5.1.2 Fragment TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //le comportement du composant PTC
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //le comportement du test élémentaire
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //envoi de données à un composant de test
        …
    }
    …
}
```

11.5.2 Scénario d'utilisation: communication internodale entre composants de test

Le scénario décrit dans la Figure 30 montre la communication entre composants de test (composant de test principal ou composants de test parallèles), qui résident dans différents nœuds. Une requête de communication est envoyée à la sous-interface TCI-CH, qui alors décide de l'endroit où il convient de mettre en file d'attente ce modèle de communication. Dans ce cas, la communication est effectuée à distance via l'exécutable TE situé dans un autre nœud. Le scénario montre une communication en mode message au moyen de l'opération d'expédition – le scénario est le même que pour les opérations suivantes: appel, réponse et propagation d'exception.

11.5.2.1 Diagramme séquentiel

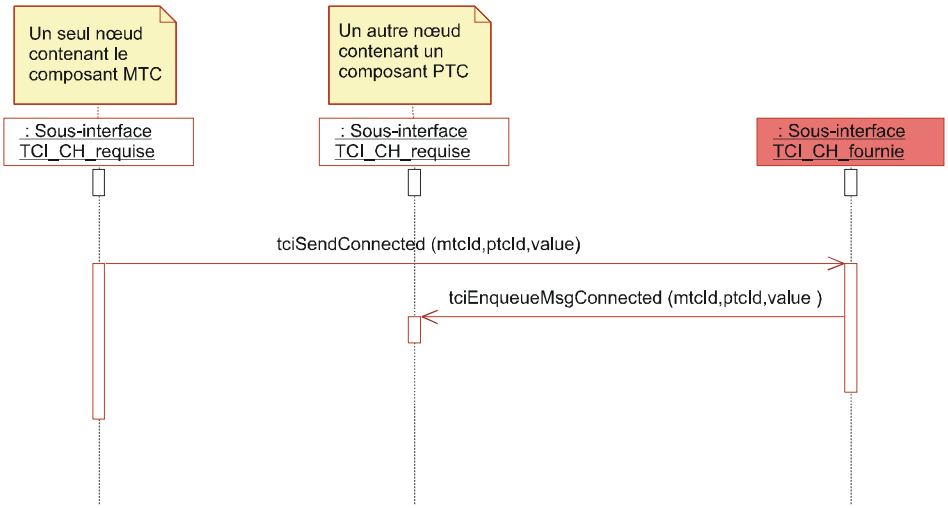

Figure 30/Z.145 – Scénario d'utilisation – Communication internodale entre composants de test

11.5.2.2 Fragment TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component ATCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    function APTCBehaviour(…) runs on APTCType {
        … //le comportement du composant PTC
    …
    }
    …
    testcase ATestCase(…) runs on ATCType… {
        … //le comportement du test élémentaire
        var ATCType PTC1:= ATCType.create;
        connect(PTC1:APort,mtc:APort);
        …
        PTC1.start(APTCBehaviour(…));
        APort.send(AMessageTemplate); //envoi de données à un composant de test
        …
    }
    …
}
```

11.5.3 Scénario d'utilisation: codage

Le scénario décrit dans la Figure 31 montre le codage de données envoyées au système SUT. Les données codées sont reçues de l'entité de codage/décodage (codec) via l'interface TCI-CD. La valeur codée est envoyée au système SUT via l'adaptateur TRI-SA. Le scénario est le même que pour les opérations d'appel, de réponse et de propagation d'exception.

11.5.3.1 Diagramme séquentiel

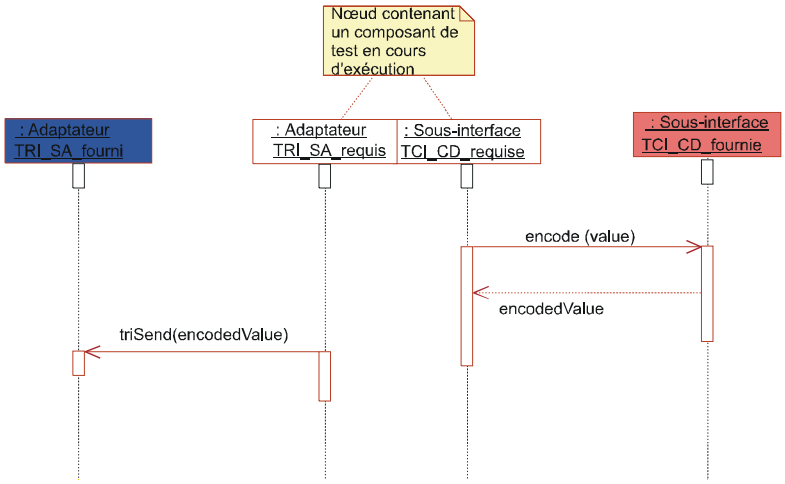

Figure 31/Z.145 – Scénario d'utilisation – Codage

11.5.3.2 Fragment TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //le comportement du test élémentaire
        map(mtc:APort,system:APort);
        …
        APort.send(AMessageTemplate); //envoi de données au système SUT
        …
    }
    …
} with { encoding = '…' }
```

11.5.4 Scénario d'utilisation: décodage

Le scénario décrit dans la Figure 32 montre le décodage de données reçues du système SUT via l'adaptateur TRI-SA. Les données décodées sont reçues de l'entité de codage/décodage (codec) via l'interface TCI-CD. Le scénario est le même que pour les opérations de réception, de traitement de réponse, d'acquisition et de vérification.

11.5.4.1 Diagramme séquentiel

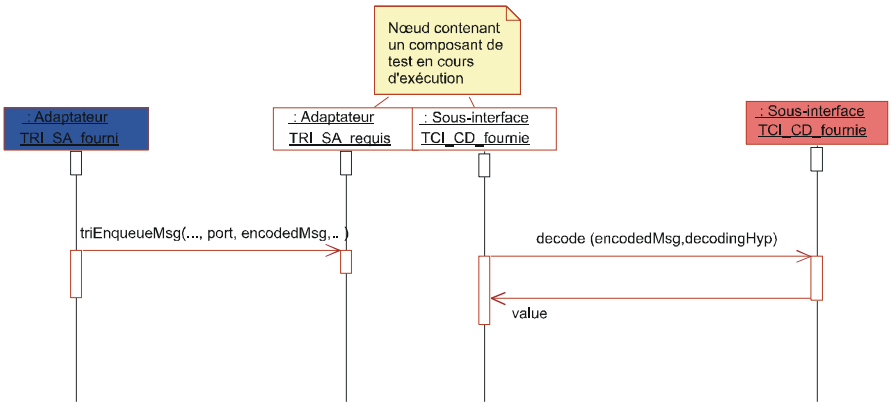

Figure 32/Z.145 – Scénario d'utilisation – Décodage

11.5.4.2 Fragment TTCN-3

```
module AModule {
    …
    type port APortType message { … }
    …
    type component APTCType {
        …
        APortType APort;
        …
    }
    …
    template AType AMessageTemplate { … }
    …
    testcase ATestCase(…) runs on APTCType system APTCType {
        … //le comportement du test élémentaire
        map(mtc:APort,system:APort);
        …
        APort.receive(AMessageTemplate); //la réception de données à partir du système SUT
        …
    }
    …
} with { encoding = '…' }
```

Annexe A

Spécification en langage IDL d'interface TCI

La présente annexe définit les interfaces de commande en notation TTCN-3 au moyen du langage de définition d'interface (IDL).

```
// ********************************************************************************
// * Définitions d'interface pour les interfaces de commande en notation TTCN-3
// ********************************************************************************

module tciInterface {

  /* Déclaration anticipée*/
  interface Value;
  interface Type;

  // ********************************************************************************
  // * Type de données extraits des définitions d'interface TRI
  // ********************************************************************************

  // Connexion
  native TriPortIdType ;
  native TriPortIdListType;
  native TriComponentIdType ;
native TriComponentIdListType;

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

  // Considérations diverses
  native TriStatusType;
  native TriTimerIdType;
native TriTimerDurationType;

  // ********************************************************************************
  // * Types généraux de données abstraites
  // ********************************************************************************

  // Définitions de base
  native TBoolean;
  native TFloat;
  native TChar;
  native TInteger;
  native TString;
  native TUniversalChar;
  typedef sequence <TString> TStringSeq;
  native TObjid;

  struct QualifiedName {
    TString moduleName;
    TString baseName;
  };

  // Types généraux de données abstraites à l'interface TCI
  typedef QualifiedName TciBehaviourIdType;
  typedef QualifiedName TciModuleIdType;
  typedef QualifiedName TciModuleParameterIdType;
  typedef QualifiedName TciTestCaseIdType;

  enum TciParameterPassingModeType {
    IN_MODE,
    OUT_MODE,
    INOUT_MODE
  };
```

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

typedef sequence <TciParameterType> TciParameterListType;

struct TciParameterTypeType {
  Type parameterType;
  TciParameterPassingModeType mode;
};

typedef sequence <TciParameterTypeType> TciParameterTypeListType;

struct TciModuleParameterType {
  TciModuleParameterIdType parameterName;
  Value defaultValue;
};

typedef sequence <TciModuleIdType> TciModuleIdListType ;

typedef sequence <TciModuleParameterType> TciModuleParameterListType;

typedef sequence <TciTestCaseIdType> TciTestCaseIdListType;


enum TciTestComponentKindType {
  MTC,
  PTC,
  CONTROL
};

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


// ***************************************************************************
// * Données abstraites en notation TTCN-3 - Types et valeurs
// ***************************************************************************

// Type de données abstraites "Type"
interface Type {
  TciModuleIdType  getDefiningModule ();
  TString          getName ();
  TciTypeClassType getTypeClass ();
  Value            newInstance ();
  TString          getTypeEncoding ();
  TString          getTypeEncodingVariant ();
  TStringSeq       getTypextension ();
};

// Valeurs abstraites TTCN-3
interface Value {
  TString  getValueEncoding ();
  TString  getValueEncodingVariant ();
  Type     getType ();
  TBoolean  notPresent ();
};
```

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

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

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

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

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

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

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

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

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

interface IntegerValue : Value {
  TInteger getInt ();
```

```
  void     setInt (in TInteger value);
};

interface CharValue : Value {
  TChar getChar ();
  void  setChar (in TChar value);
};

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

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

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

};

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

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

};

// ***************************************************************************
// * Types de journalisation abstraite
// ***************************************************************************

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

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

typedef sequence <Value> TciValueList;

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

```
typedef sequence <TciValueDifference> TciValueDifferenceList;


// ********************************************************************************
// Interface de codage-décodage
// -  Requise
// ********************************************************************************

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

// ********************************************************************************
// Interface de codage-décodage
// -  Fournie
// ********************************************************************************

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


// ********************************************************************************
// Interface de gestion de test
// -  Requise
// ********************************************************************************

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

// ********************************************************************************
// Interface de gestion de test
// -  Fournie
// ********************************************************************************

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

```
  Value tciGetModulePar (
          in TciModuleParameterIdType parameterId
          );
  void tciLog (
       in TriComponentIdType testComponentId,
       in TString message
       );
  void tciError (in TString message);
};


// ********************************************************************************
// Interface de traitement de composant
// -  Requise
// ********************************************************************************

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

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

// *****************************************************************************
// Interface de traitement de composant
// -  Fournie
// *****************************************************************************

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

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

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

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

```
        in TriPortIdType sender,
        in TriComponentIdListType receivers,
        in TriSignatureIdType signature,
        in Value except
        );

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

// ***************************************************************************
// Interface de journalisation de test
// -  Fournie
// ***************************************************************************

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

```
       in TriComponentIdType c, in TciTestCaseIdType tcId,
       in TriParameterListType pars, in VerdictValue outcome);

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

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

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

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

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

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

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

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

```
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in TciValueTemplate parsTmpl, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );

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

void tliPrReply_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TriComponentIdType to,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_c_BC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriStatusType transmissionFailure
        );
 void tliPrReply_c_MC(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value parsValue, in Value replValue,
        in TriComponentIdListType toList, in TriStatusType transmissionFailure
        );

void tliPrGetReplyDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TriParameterType repl,
        in TriAddressType address
        );
void tliPrGetReplyDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value replValue,
        in TriComponentIdType from
        );
void tliPrGetReplyMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
        in TciValueDifferenceList diffs, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
void tliPrGetReplyMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value replValue, in TciValueTemplate replyTmpl,
        in TciValueDifferenceList diffs, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );
void tliPrGetReply_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
```

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

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

    void tliPrRaise_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TriComponentIdType to,
        in TriStatusType transmissionFailure
        );
    void tliPrCatchDetected_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TriExceptionType exc,
        in TriAddressType address
        );
    void tliPrCatchDetected_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in Value excValue,
        in TriComponentIdType from
        );
    void tliPrCatchMismatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriAddressType address,
        in TciValueTemplate addressTmpl
        );
    void tliPrCatchMismatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TciValueDifferenceList diffs, in TriComponentIdType from,
        in TciNonValueTemplate fromTmpl
        );
    void tliPrCatch_m(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
        in TriSignatureIdType signature, in TciParameterListType parsValue,
        in Value excValue, in TciValueTemplate excTmpl,
        in TriAddressType address, in TciValueTemplate addressTmpl);

    void tliPrCatch_c(
        in TString am, in TInteger ts, in TString src, in TInteger line,
        in TriComponentIdType c, in TriPortIdType port,
```

```
       in TriSignatureIdType signature, in TciParameterListType parsValue,
       in Value excValue, in TciValueTemplate excTmpl,
       in TriComponentIdType from, in TciNonValueTemplate fromTmpl
       );
void tliPrCatchTimeoutDetected(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port,
       in TriSignatureIdType signature
       );
void tliPrCatchTimeout(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port,
       in TriSignatureIdType signature, in TciParameterListType parsValue
       );
void tliCCreate(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp,
       in String name
       );
void tliCStart(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp,
       in TciBehaviourIdType name, in TciParameterListType parsValue
       );
void tliCRunning(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType comp, in TBoolean status
       );
 void tliCAlive(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c,
         in TriComponentIdType comp, in TBoolean status
         );
 void tliCStop(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCKill(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TriComponentIdType comp
         );
 void tliCDoneMismatch(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCKilledMismatch(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCDone(in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
 void tliCKilled(
         in TString am, in TInteger ts, in TString src, in TInteger line,
         in TriComponentIdType c, in TciNonValueTemplate compTmpl
         );
void tliCTerminated(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in VerdictValue verdict
       );
void tliPConnect(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPDisconnect(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPMap(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
void tliPUnmap(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriComponentIdType c1, in TriPortIdType port1,
       in TriComponentIdType c2, in TriPortIdType port2
       );
```

```
void tliPClear(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port
       );
void tliPStart(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port
       );
void tliPStop(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port
       );
void tliPHalt(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriPortIdType port
       );
void tliEncode(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in Value val, in TriStatusType encoderFailure,
       in TriMessageType msg, in TString codec
       );
void tliDecode(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriMessageType msg,
       in TriStatusType decoderFailure, in Value val, in TString codec
       );
void tliTTimeoutDetected(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriTimerIdType timer
       );
void tliTTimeoutMismatch(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TciNonValueTemplate timerTmpl
       );
void tliTTimeout(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TciNonValueTemplate timerTmpl
       );
void tliTStart(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriTimerIdType timer,
       in TriTimerDurationType dur
       );
void tliTStop(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriTimerIdType timer
       );
void tliTRead(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriTimerIdType timer,
       in TriTimerDurationType elapsed
       );
void tliTRunning(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TriTimerIdType timer, in TBoolean status
       );
void tliSEnter(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TString name, in TciParameterListType parsValue,
       in TString kind
       );
void tliSLeave(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TString name, in Value returnValue,
       in TString kind
       );
void tliVar(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TString name, in Value varValue
       );
void tliModulePar(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in TString name, in Value parValue
       );
void tliGetVerdict(
       in TString am, in TInteger ts, in TString src, in TInteger line,
       in TriComponentIdType c, in VerdictValue verdict
       );
void tliSetVerdict(
       in TString am, in TInteger ts, in TString src, in TInteger line,
```

```
            in TriComponentIdType c, in VerdictValue verdict
            );
        void tliLog(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TciValueList log
            );
        void tliAEnter(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
        void tliALeave(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
        void tliADefaults(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
        void tliAActivate(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in TString name, in TciParameterListType pars,
            in Value ref
            );
        void tliADeactivate(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c, in Value ref
            );
        void tliANomatch(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
        void tliARepeat(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
        void tliAWait(
            in TString am, in TInteger ts, in TString src, in TInteger line,
            in TriComponentIdType c
            );
    };
};
```

Annexe B

Mappage vers le langage XML pour sous-interface TCI-TL fournie

La présente annexe définit un mappage pour l'interface de journalisation d'interface TCI utilisant les définitions de schéma du langage de balisage étendu (XML, eXtended Markup Language).

B.1 Schéma de mappage vers le langage XML d'une interface TCI-TL pour types simples

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
            xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
            elementFormDefault="qualified">

    <!-- Définitions de base -->
    <xsd:simpleType name="xpath">
        <!-- Cette chaîne devrait être conforme à XPATH -->
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TBoolean">
        <xsd:restriction base="xsd:boolean"/>
    </xsd:simpleType>

    <xsd:simpleType name="TString">
        <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>

    <xsd:simpleType name="TInteger">
        <xsd:restriction base="xsd:integer"/>
    </xsd:simpleType>

    <!-- Considérations diverses -->
    <xsd:simpleType name="TriTimerDurationType">
        <xsd:restriction base="xsd:float"/>
    </xsd:simpleType>

    <xsd:simpleType name="TciParameterPassingModeType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="in"/>
            <xsd:enumeration value="inout"/>
            <xsd:enumeration value="out"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TriStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TRI_Ok"/>
            <xsd:enumeration value="TRI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="TciStatusType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="TCI_Ok"/>
            <xsd:enumeration value="TCI_Error"/>
        </xsd:restriction>
    </xsd:simpleType>

</xsd:schema>
```

B.2 Schéma de mappage vers le langage XML d'une interface TCI-TL pour types

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
        elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
```

```
 schemaLocation="SimpleTypes.xsd"/>

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

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

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

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

<!-- Communication -->
<xsd:complexType name="TriMessageType">
    <xsd:attribute name="val" type="xsd:hexBinary"/>
</xsd:complexType>

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

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

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

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

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

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


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

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

    <!-- Définitions de base -->
    <xsd:complexType name="QualifiedName">
        <xsd:attribute name="moduleName" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="baseName" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- Types généraux de données abstraites à l'interface TCI -->
    <xsd:complexType name="TciBehaviourIdType">
        <xsd:sequence>
            <xsd:element name="name" type="Types:QualifiedName" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

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

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

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

    <!-- Structure générale des identificateurs pour composants de test, ports et temporisateurs -->
    <xsd:complexType name="Id">
        <xsd:sequence>
           <xsd:element name="name" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="id" type="SimpleTypes:TInteger" minOccurs="1" maxOccurs="1"/>
           <xsd:element name="type" type="SimpleTypes:TString" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

B.3 Schéma de mappage vers le langage XML d'une interface TCI-TL pour valeurs

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates.xsd"
     schemaLocation="Templates.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
     schemaLocation="SimpleTypes.xsd"/>

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

    <xsd:complexType name="Value" mixed="true">
        <xsd:choice>
            <xsd:element name="integer" type="Values:IntegerValue"/>
            <xsd:element name="float" type="Values:FloatValue"/>
            <xsd:element name="boolean" type="Values:BooleanValue"/>
            <xsd:element name="objid" type="Values:ObjidValue"/>
            <xsd:element name="verdicttype" type="Values:VerdictValue"/>
```

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

<!-- éléments événementiels généraux -->
<xsd:complexType name="IntegerValue">
    <xsd:simpleContent>
        <xsd:extension base="SimpleTypes:TString">
                <xsd:attributeGroup ref="Values:ValueAtts"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

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

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

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

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

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

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

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

```
</xsd:complexType>

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

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

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

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

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

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

<xsd:complexType name="SetOfValue">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="integer" type="Values:IntegerValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="float" type="Values:FloatValue" minOccurs="0"
                maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:sequence>
            <xsd:element name="boolean" type="Values:BooleanValue" minOccurs="0"
                maxOccurs="unbounded"/>
```

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

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

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

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

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

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

B.4 Schéma de mappage vers le langage XML d'une interface TCI-TL pour modèles

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
```

```
     schemaLocation="SimpleTypes.xsd"/>

    <xsd:complexType name="TciValueTemplate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Values:Value">
            <xsd:choice>
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                <xsd:element name="float" type="Templates:FloatTemplate"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                <xsd:element name="universal_charstring"
type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
            <xsd:element name="union" type="Templates:UnionTemplate"/>
            <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
            <xsd:element name="address" type="Templates:AddressTemplate"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
             </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

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

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

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


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

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


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

```
            <xsd:attribute name="tmpl" type="SimpleTypes:xpath" use="required"/>
    </xsd:complexType>

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


    <xsd:complexType name="IntegerTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:IntegerValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="FloatTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:FloatValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="BooleanTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:BooleanValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="ObjidTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:ObjidValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="BitstringTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:BitstringValue">
            <xsd:choice>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="null" type="xsd:string"/>
            </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
```

```
<xsd:complexType name="HexstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:BitstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="OctetstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:OctetstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CharstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:CharstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UniversalCharstringTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:UniversalCharstringValue">
        <xsd:choice>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="null" type="xsd:string"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RecordTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:RecordValue">
            <xsd:sequence>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
```

```
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                    <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                    </xsd:choice>
                </xsd:sequence>
    </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RecordOfTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:RecordOfValue">
            <xsd:choice>
                <xsd:sequence>
                    <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
```

```
                    <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:SetValue">
            <xsd:sequence>
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SetOfTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:SetOfValue">
            <xsd:choice>
                <xsd:sequence>
                    <xsd:element name="integer" type="Templates:IntegerTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="float" type="Templates:FloatTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="objid" type="Templates:ObjidTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
```

```
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record" type="Templates:RecordTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="record_of" type="Templates:RecordOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set" type="Templates:SetTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="set_of" type="Templates:SetOfTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"
                        minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="union" type="Templates:UnionTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="anytype" type="Templates:AnytypeTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                <xsd:sequence>
                    <xsd:element name="address" type="Templates:AddressTemplate" minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
            </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="EnumeratedTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:EnumeratedValue">
            <xsd:choice>
                    <xsd:element name="omit" type="Templates:omit"/>
                    <xsd:element name="any" type="Templates:any"/>
                    <xsd:element name="anyoromit" type="Templates:anyoromit"/>
            <xsd:element name="templateDef" type="SimpleTypes:TString"/>
        </xsd:choice>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="UnionTemplate">
    <xsd:complexContent>
        <xsd:extension base="Values:UnionValue">
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                <xsd:element name="float" type="Templates:FloatTemplate"/>
                <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                <xsd:element name="universal_charstring"
                    type="Templates:UniversalCharstringTemplate"/>
            <xsd:element name="record" type="Templates:RecordTemplate"/>
            <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
            <xsd:element name="set" type="Templates:SetTemplate"/>
            <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
            <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
            <xsd:element name="union" type="Templates:UnionTemplate"/>
```

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

    <xsd:complexType name="AnytypeTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="address" type="Templates:AddressTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="AddressTemplate">
        <xsd:complexContent>
            <xsd:extension base="Values:AnytypeValue">
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                    <xsd:element name="integer" type="Templates:IntegerTemplate"/>
                    <xsd:element name="float" type="Templates:FloatTemplate"/>
                    <xsd:element name="boolean" type="Templates:BooleanTemplate"/>
                    <xsd:element name="objid" type="Templates:ObjidTemplate"/>
                    <xsd:element name="bitstring" type="Templates:BitstringTemplate"/>
                    <xsd:element name="hexstring" type="Templates:HexstringTemplate"/>
                    <xsd:element name="octetstring" type="Templates:OctetstringTemplate"/>
                    <xsd:element name="charstring" type="Templates:CharstringTemplate"/>
                    <xsd:element name="universal_charstring"
                        type="Templates:UniversalCharstringTemplate"/>
                <xsd:element name="record" type="Templates:RecordTemplate"/>
                <xsd:element name="record_of" type="Templates:RecordOfTemplate"/>
                <xsd:element name="set" type="Templates:SetTemplate"/>
                <xsd:element name="set_of" type="Templates:SetOfTemplate"/>
                <xsd:element name="enumerated" type="Templates:EnumeratedTemplate"/>
                <xsd:element name="union" type="Templates:UnionTemplate"/>
                <xsd:element name="anytype" type="Templates:AnytypeTemplate"/>
                        <xsd:element name="omit" type="Templates:omit"/>
                        <xsd:element name="any" type="Templates:any"/>
                        <xsd:element name="anyoromit" type="Templates:anyoromit"/>
                <xsd:element name="templateDef" type="SimpleTypes:TString"/>
                </xsd:choice>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

B.5 Schéma de mappage vers le langage XML d'une interface TCI-TL pour événements

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events"
    xmlns:Events="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
```

```
    xmlns:Templates="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates"
    xmlns:SimpleTypes="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values" elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/SimpleTypes.xsd"
     schemaLocation="SimpleTypes.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Templates.xsd"
     schemaLocation="Templates.xsd"/>

    <!-- définition commune pour tous événements -->
    <xsd:complexType name="Event" mixed="true">
        <xsd:sequence>
            <xsd:element name="am" type="SimpleTypes:TString"/>
        </xsd:sequence>
        <xsd:attribute name="ts" type="xsd:time" use="required"/>
        <xsd:attribute name="src" type="SimpleTypes:TString" use="optional"/>
        <xsd:attribute name="line" type="SimpleTypes:TInteger" use="optional"/>

        <!-- structure générale des identificateurs pour composants de test, ports et
        <!-- temporisateurs -->
        <xsd:attribute name="name" type="SimpleTypes:TString" use="required"/>
        <xsd:attribute name="id" type="SimpleTypes:TInteger" use="required"/>
        <xsd:attribute name="type" type="SimpleTypes:TString" use="required"/>
    </xsd:complexType>

    <!-- cet événement est étendu par tous les événements de configuration de port -->
    <xsd:complexType name="PortConfiguration">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port1" type="Types:TriPortIdType" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="port2" type="Types:TriPortIdType" minOccurs="1"
maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- cet événement est étendu par tous les événements de description d'état de port -->
    <xsd:complexType name="PortStatus">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- Tests élémentaires -->
    <xsd:complexType name="tliTcExecute">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcStart">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcStop">
```

```
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcStarted">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="dur" type="Types:TriTimerDurationType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTcTerminated">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="tcId" type="Types:TciTestCaseIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="outcome" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- commande -->
    <xsd:complexType name="tliCtrlStart">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCtrlStop">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliCtrlTerminated">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- communication asynchrone -->
    <xsd:complexType name="tliMSend_m">
         <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_m_BC">
         <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
```

```
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_m_MC">
         <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="msg" type="Types:TriMessageType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMSend_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Values:Value"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="msgValue" type="Types:TriMessageType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
```

```
    </xsd:complexType>

    <xsd:complexType name="tliMDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Types:TriMessageType"/>
                        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMMismatch_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Values:Value"/>
                        <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Values:Value"/>
                        <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"/>
                        <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMReceive_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                        <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                        <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                        <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliMReceive_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                    <xsd:sequence>
                        <xsd:element name="port" type="Types:TriPortIdType"/>
                        <xsd:element name="msgValue" type="Values:Value" minOccurs="0"/>
                        <xsd:element name="msgTmpl" type="Templates:TciValueTemplate"
minOccurs="0"/>
                        <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                        <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"
minOccurs="0"/>
                    </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

     <!-- communication synchrone -->
    <xsd:complexType name="tliPrCall_m">
```

```
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"
minOccurs="0"/>
                        <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
```

```
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrCall_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType" minOccurs="0"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcallDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcallDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcallMismatch_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcallMismatch_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcall_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
```

```
                    <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TriParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetcall_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="from" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                    <xsd:element name="pars" type="Types:TciParameterListType"/>
                    <xsd:element name="parsTmpl" type="Templates:TciValueTemplate"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
```

```
                            <xsd:element name="repl" type="Types:TriParameterType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrReply_c_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyDetected_m">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="address" type="Types:TriAddressType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrGetReplyDetected_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="replValue" type="Values:Value"/>
                    <xsd:element name="from" type="Types:TriComponentIdType"/>
                </xsd:sequence>
            </xsd:extension>
```

```
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                <xsd:element name="replValue" type="Values:Value"/>
                <xsd:element name="replTmpl" type="Values:Value"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="address" type="Types:TriAddressType"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReplyMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="replValue" type="Values:Value"/>
                <xsd:element name="replTmpl" type="Values:Value"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="from" type="Types:TriComponentIdType"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                <xsd:element name="replValue" type="Values:Value"/>
                <xsd:element name="replTmpl" type="Values:Value"/>
                <xsd:element name="address" type="Types:TriAddressType"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrGetReply_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="replValue" type="Values:Value"/>
                <xsd:element name="replTmpl" type="Values:Value"/>
                <xsd:element name="from" type="Types:TriComponentIdType"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
                <xsd:choice>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
```

```
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_m_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_m_MC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TriParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="addresses" type="Types:TriAddressListType" minOccurs="0"/>
                    <xsd:choice>
                        <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                        <xsd:sequence>
                            <xsd:element name="exc" type="Types:TriExceptionType" minOccurs="0"/>
                            <xsd:element name="transmission-failure"
type="SimpleTypes:TriStatusType" minOccurs="0"/>
                        </xsd:sequence>
                    </xsd:choice>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_c">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="to" type="Types:TriComponentIdType" minOccurs="0"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliPrRaise_c_BC">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="port" type="Types:TriPortIdType"/>
                    <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                    <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                    <xsd:element name="excValue" type="Values:Value"/>
                    <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
```

```
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrRaise_c_MC">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="excValue" type="Values:Value"/>
                <xsd:element name="toList" type="Types:TriComponentIdListType" minOccurs="0"/>
                <xsd:element name="transmission-failure" type="SimpleTypes:TriStatusType"
minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="exc" type="Types:TriExceptionType"/>
                <xsd:element name="address" type="Types:TriAddressType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchDetected_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="exc" type="Types:TriExceptionType"/>
                <xsd:element name="from" type="Types:TriComponentIdType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="exc" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="diffs" type="Templates:TciValueDifferenceList"/>
                <xsd:element name="address" type="Types:TriAddressType"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchMismatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TciParameterListType"/>
                <xsd:element name="exc" type="Values:Value"/>
                <xsd:element name="excTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="address" type="Types:TriAddressType"/>
                <xsd:element name="addressTmpl" type="Templates:TciValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:complexType name="tliPrCatch_m">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="exception" type="Values:Value"/>
                <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatch_c">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="signatureTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="exception" type="Values:Value"/>
                <xsd:element name="exceptionTmpl" type="Templates:TciValueTemplate"/>
                <xsd:element name="from" type="Types:TriComponentIdType"/>
                <xsd:element name="fromTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeoutDetected">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPrCatchTimeout">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="port" type="Types:TriPortIdType"/>
                <xsd:element name="signature" type="Types:TriSignatureIdType"/>
                <xsd:element name="parsValue" type="Types:TriParameterListType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- composants -->
<xsd:complexType name="tliCCreate">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="name" type="SimpleTypes:TString"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStart">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="name" type="Types:TciBehaviourIdType"/>
                <xsd:element name="pars" type="Types:TciParameterListType" minOccurs="0"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCRunning">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
```

```
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="status" type="SimpleTypes:TBoolean"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>


<xsd:complexType name="tliCAlive">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="status" type="SimpleTypes:TBoolean"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCStop">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKill">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDoneMismatch">
   <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
            <xsd:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCKilledMismatch">
   <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
            <xsd:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCDone">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
            <xsd:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

 <xsd:complexType name="tliCKilled">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
```

```
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="compTmpl" type="Templates:TciNonValueTemplate"/>
            </xsd:sequence>
            <xsd:attribute name="done" type="SimpleTypes:TBoolean"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliCTerminated">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="comp" type="Types:TriComponentIdType"/>
                <xsd:element name="verdict" type="Values:VerdictValue" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- ports -->
<xsd:complexType name="tliPConnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPDisconnect">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPMap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPUnmap">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPClear">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStart">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPStop">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="tliPHalt">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:PortConfiguration"/>
    </xsd:complexContent>
</xsd:complexType>

<!-- codec -->
<xsd:complexType name="tliEncode">
    <xsd:complexContent mixed="true">
        <xsd:extension base="Events:Event">
            <xsd:sequence>
                <xsd:element name="val" type="Values:Value"/>
                <xsd:choice>
                    <xsd:element name="msg" type="Types:TriMessageType"/>
                    <xsd:element name="encoder-failure" type="SimpleTypes:TciStatusType"/>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/>
```

```
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliDecode" mixed="true">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:choice>
                        <xsd:element name="val" type="Values:Value"/>
                        <xsd:element name="decoder-failure" type="SimpleTypes:TciStatusType"/>
                    </xsd:choice>
                    <xsd:element name="msg" type="Types:TriMessageType"/>
                </xsd:sequence>
                <xsd:attribute name="codec" type="SimpleTypes:TString" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- temporisateurs -->
    <xsd:complexType name="tliTTimeoutDetected">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1"
minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTTimeoutMismatch">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1"
minOccurs="1"/>
                    <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1"
minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTTimeout">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType" maxOccurs="1"
minOccurs="1"/>
                    <xsd:element name="timerTmpl" type="Templates:TciNonValueTemplate" maxOccurs="1"
minOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTStart">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType"/>
                    <xsd:element name="dur" type="Types:TriTimerDurationType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTStop">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTRead">
        <xsd:complexContent mixed="true">
```

```
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType"/>
                    <xsd:element name="elapsed" type="Types:TriTimerDurationType"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliTRunning">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="timer" type="Types:TriTimerIdType"/>
                </xsd:sequence>
                <xsd:attribute name="status" type="SimpleTypes:TBoolean"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- portée -->
    <xsd:complexType name="tliSEnter">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="kind" type="SimpleTypes:TString"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliSLeave">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="return" type="Values:Value" minOccurs="0"/>
                    <xsd:element name="kind" type="SimpleTypes:TString"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- variables et paramètres de module -->
    <xsd:complexType name="tliVar">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="val" type="Values:Value" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliModuleParr">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="val" type="Values:Value" minOccurs="0"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- verdicts -->
    <xsd:complexType name="tliGetVerdict">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="verdict" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
```

```
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliSetVerdict">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="verdict" type="Values:VerdictValue"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- journalisation -->
    <xsd:complexType name="tliLog">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="log" type="SimpleTypes:TString"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <!-- instruction "alt" (comportement à options) -->
    <xsd:complexType name="tliAEnter">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliALeave">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliADefaults">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliAActivate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="name" type="Types:QualifiedName" minOccurs="1"
maxOccurs="1"/>
                    <xsd:element name="pars" type="Types:TriParameterListType" minOccurs="0"/>
                    <xsd:element name="ref" type="Values:Value"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliADeactivate">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event">
                <xsd:sequence>
                    <xsd:element name="ref" type="Values:Value"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliANomatch">
        <xsd:complexContent mixed="true">
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliARepeat">
        <xsd:complexContent>
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="tliAWait">
        <xsd:complexContent>
```

```
            <xsd:extension base="Events:Event"/>
        </xsd:complexContent>
    </xsd:complexType>

</xsd:schema>
```

B.6 Schéma de mappage vers le langage XML d'une interface TCI-TL pour un journal

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI"
    xmlns:TLI="http://uri.etsi.org/ttcn-3/3.0.0/tci/TLI"
    xmlns:Types="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types"
    xmlns:Values="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values"
    xmlns:Events="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events" elementFormDefault="qualified">

    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Types.xsd"
schemaLocation="Types.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Values.xsd"
     schemaLocation="Values.xsd"/>
    <xsd:import namespace="http://uri.etsi.org/ttcn-3/3.0.0/tci/Events.xsd"
     schemaLocation="Events.xsd"/>

    <xsd:element name="logfile" type="TLI:LogModule"/>
    <xsd:complexType name="LogModule">
        <xsd:sequence>
            <xsd:element name="header" type="TLI:Header"/>
            <xsd:element name="body" type="TLI:Body"/>
            <xsd:element name="trailer" type="TLI:Trailer"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Header">
        <xsd:sequence>
           <!-- version de la journalisation -->
           <xsd:element name="version" type="xsd:string"/>
          <!-- début du fichier de journalisation -->
            <xsd:element name="ts" type="xsd:time"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="Trailer">
        <xsd:sequence/>
    </xsd:complexType>

    <xsd:complexType name="Body">
        <xsd:choice maxOccurs="unbounded">

            <!-- opérations de test élémentaire -->
            <xsd:element name="tliTcExecute" type="Events:tliTcExecute"/>
            <xsd:element name="tliTcStart" type="Events:tliTcStart"/>
            <xsd:element name="tliTcStop" type="Events:tliTcStop"/>
            <xsd:element name="tliTcStarted" type="Events:tliTcStarted"/>
            <xsd:element name="tliTcTerminated" type="Events:tliTcTerminated"/>

            <!-- opérations de commande -->
            <xsd:element name="tliCtrlStart" type="Events:tliCtrlStart"/>
            <xsd:element name="tliCtrlStop" type="Events:tliCtrlStop"/>
            <xsd:element name="tliCtrlTerminated" type="Events:tliCtrlTerminated"/>

            <!-- communication asynchrone -->
            <xsd:element name="tliMSend_m" type="Events:tliMSend_m"/>
            <xsd:element name="tliMSend_c" type="Events:tliMSend_c"/>
            <xsd:element name="tliMDetected_m" type="Events:tliMDetected_m"/>
            <xsd:element name="tliMDetected_c" type="Events:tliMDetected_c"/>
            <xsd:element name="tliMMismatch_m" type="Events:tliMMismatch_m"/>
            <xsd:element name="tliMMismatch_c" type="Events:tliMMismatch_c"/>
            <xsd:element name="tliMReceive_m" type="Events:tliMReceive_m"/>
            <xsd:element name="tliMReceive_c" type="Events:tliMReceive_c"/>

            <!-- communication synchrone -->
            <xsd:element name="tliPrCall_m" type="Events:tliPrCall_m"/>
            <xsd:element name="tliPrCall_c" type="Events:tliPrCall_c"/>

            <xsd:element name="tliPrGetcallDetected_m" type="Events:tliPrGetcallDetected_m"/>
            <xsd:element name="tliPrGetcallDetected_c" type="Events:tliPrGetcallDetected_c"/>
            <xsd:element name="tliPrGetcallMismatch_m" type="Events:tliPrGetcallMismatch_m"/>
            <xsd:element name="tliPrGetcallMismatch_c" type="Events:tliPrGetcallMismatch_c"/>
            <xsd:element name="tliPrGetcall_m"     type="Events:tliPrGetcall_m"/>
            <xsd:element name="tliPrGetcall_c"     type="Events:tliPrGetcall_c"/>

            <xsd:element name="tliPrReply_m"    type="Events:tliPrReply_m"/>
```

```
            <xsd:element name="tliPrReply_c"    type="Events:tliPrReply_c"/>

            <xsd:element name="tliPrGetReplyDetected_m" type="Events:tliPrGetReplyDetected_m"/>
            <xsd:element name="tliPrGetReplyDetected_c" type="Events:tliPrGetReplyDetected_c"/>
            <xsd:element name="tliPrGetReplyMismatch_m" type="Events:tliPrGetReplyMismatch_m"/>
            <xsd:element name="tliPrGetReplyMismatch_c" type="Events:tliPrGetReplyMismatch_c"/>
            <xsd:element name="tliPrGetReply_m"     type="Events:tliPrGetReply_m"/>
            <xsd:element name="tliPrGetReply_c"     type="Events:tliPrGetReply_c"/>

            <xsd:element name="tliPrRaise_m"    type="Events:tliPrRaise_m"/>
            <xsd:element name="tliPrRaise_c"    type="Events:tliPrRaise_c"/>

            <xsd:element name="tliPrCatchDetected_m"    type="Events:tliPrCatchDetected_m"/>
            <xsd:element name="tliPrCatchDetected_c"    type="Events:tliPrCatchDetected_c"/>
            <xsd:element name="tliPrCatchMismatch_m"    type="Events:tliPrCatchMismatch_m"/>
            <xsd:element name="tliPrCatchMismatch_c"    type="Events:tliPrCatchMismatch_c"/>
            <xsd:element name="tliPrCatch_m"    type="Events:tliPrCatch_m"/>
            <xsd:element name="tliPrCatch_c"    type="Events:tliPrCatch_c"/>

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

            <!-- composants -->
            <xsd:element name="tliCCreate" type="Events:tliCCreate"/>
            <xsd:element name="tliCStart" type="Events:tliCStart"/>
            <xsd:element name="tliCRunning" type="Events:tliCRunning"/>
            <xsd:element name="tliCAlive" type="Events:tliCRunning"/>
            <xsd:element name="tliCStop" type="Events:tliCStop"/>
            <xsd:element name="tliCKill" type="Events:tliCStop"/>
            <xsd:element name="tliCDoneMismatch" type="Events:tliCDone"/>
            <xsd:element name="tliCDone" type="Events:tliCDone"/>
            <xsd:element name="tliCKilledMismatch" type="Events:tliCDone"/>
            <xsd:element name="tliCKilled" type="Events:tliCDone"/>
            <xsd:element name="tliCTerminated" type="Events:tliCTerminated"/>

            <!-- ports -->
            <xsd:element name="tliPConnect" type="Events:tliPConnect"/>
            <xsd:element name="tliPDisconnect" type="Events:tliPDisconnect"/>
            <xsd:element name="tliPMap" type="Events:tliPMap"/>
            <xsd:element name="tliPUnmap" type="Events:tliPUnmap"/>
            <xsd:element name="tliPClear" type="Events:tliPClear"/>
            <xsd:element name="tliPStart" type="Events:tliPStart"/>
            <xsd:element name="tliPStop" type="Events:tliPStop"/>
            <xsd:element name="tliPHalt" type="Events:tliPStop"/>

            <!-- codec -->
            <xsd:element name="tliDecode" type="Events:tliDecode"/>
            <xsd:element name="tliEncode" type="Events:tliEncode"/>

            <!-- temporisateurs -->
            <xsd:element name="tliTTimeoutDetected" type="Events:tliTTimeoutDetected"/>
            <xsd:element name="tliTTimeoutMimatch" type="Events:tliTTimeoutMismatch"/>
            <xsd:element name="tliTTimeout" type="Events:tliTTimeout"/>
            <xsd:element name="tliTStart" type="Events:tliTStart"/>
            <xsd:element name="tliTStop" type="Events:tliTStop"/>
            <xsd:element name="tliTRead" type="Events:tliTRead"/>
            <xsd:element name="tliTRunning" type="Events:tliTRunning"/>

            <!-- domaines d'application -->
            <xsd:element name="tliSEnter" type="Events:tliSEnter"/>
            <xsd:element name="tliSLeave" type="Events:tliSLeave"/>

            <!-- instructions -->
            <xsd:element name="tliVar" type="Events:tliVar"/>
            <xsd:element name="tliGetVerdict" type="Events:tliGetVerdict"/>
            <xsd:element name="tliSetVerdict" type="Events:tliSetVerdict"/>
            <xsd:element name="tliLog" type="Events:tliLog"/>

            <!-- instruction "alt" (comportement à options)  -->
            <xsd:element name="tliAEnter" type="Events:tliAEnter"/>
            <xsd:element name="tliALeave" type="Events:tliALeave"/>
            <xsd:element name="tliADefaults" type="Events:tliADefaults"/>
            <xsd:element name="tliAActivate" type="Events:tliAActivate"/>
            <xsd:element name="tliADeactivate" type="Events:tliADeactivate"/>
            <xsd:element name="tliANomatch" type="Events:tliANomatch"/>
            <xsd:element name="tliARepeat" type="Events:tliARepeat"/>
            <xsd:element name="tliAWait" type="Events:tliAWait"/>

        </xsd:choice>
    </xsd:complexType>
</xsd:schema>
```

BIBLIOGRAPHIE

- INTOOL CGI/NPL038 (V2.2): *Generic Compiler/Interpreter interface; GCI Interface Specification* (Interface générique de compilateur/interpréteur – Spécification de l'interface GCI) *Infrastructural Tools for Informational Technology and Telcomunications Conference Testing*, décembre 1996.
- Recommandation UIT-T X.292 (2002), *Cadre et méthodologie des tests de conformité OSI pour les Recommandations sur les protocoles pour les applications de l'UIT-T – Notation combinée arborescente et tabulaire (TTCN).*
- ISO/CEI 9646-3 (1998): *Technologies de l'information – Interconnexion de systèmes ouverts – Essais de conformité – Méthodologie générale et procédures – Partie 3: Notation combinée, arborescente et tabulaire (TTCN).*
- ISO/CEI 10646:2003, *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC).*
- OMG CORBA v2.2: *The Common Object Request Broker*: *Architecture and Specification* (Courtier commun de requêtes d'objets: architecture et spécification), Section 3, février 1998.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	Gestion des télécommunications y compris le RGT et maintenance des réseaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données, communication entre systèmes ouverts et sécurité
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de prochaine génération
Série Z	**Langages et aspects généraux logiciels des systèmes de télécommunication**

Imprimé en Suisse
Genève, 2007