МЕЖДУНАРОДНЫЙ СОЮЗ ЭЛЕКТРОСВЯЗИ

МСЭ-Т Z.144

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

(03/2006)

СЕРИЯ Z: ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

Методы формального описания (FDT) – Нотация тестирования и управления тестом (TTCN)

Нотация тестирования и управления тестом версии 3 (TTCN-3): Интерфейс времени выполнения (TRI)

Рекомендация МСЭ-Т Z.144

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ Z

ЯЗЫКИ И ОБЩИЕ АСПЕКТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИСТЕМ ЭЛЕКТРОСВЯЗИ

МЕТОДЫ ФОРМАЛЬНОГО ОПИСАНИЯ (FDT)	Z.100–Z.199
Язык спецификации и описания (SDL)	Z.100–Z.109
Применение методов формального описания	Z.110–Z.119
Диаграмма последовательности сообщений (MSC)	Z.120–Z.129
Расширенный язык описания объектов (eODL)	Z.130–Z.139
Нотация тестирования и управления тестированием (TTCN)	**Z.140–Z.149**
Нотация требований пользователя (URN)	Z.150–Z.159
ЯЗЫКИ ПРОГРАММИРОВАНИЯ	Z.200–Z.299
CHILL: язык высокого уровня МСЭ-Т	Z.200–Z.209
ЯЗЫК "ЧЕЛОВЕК–МАШИНА"	Z.300–Z.399
Общие принципы	Z.300–Z.309
Базисный синтаксис и диалоговые процедуры	Z.310–Z.319
Расширенный язык MML для видеотерминалов	Z.320–Z.329
Спецификация интерфейса "человек–машина"	Z.330–Z.349
Информационно-ориентированные интерфейсы "человек–машина"	Z.350–Z.359
Интерфейсы "человек–машина" для управления сетями электросвязи	Z.360–Z.379
КАЧЕСТВО	Z.400–Z.499
Качество программного обеспечения электросвязи	Z.400–Z.409
Аспекты качества рекомендаций, относящихся к протоколам	Z.450–Z.459
МЕТОДЫ	Z.500–Z.599
Методы проверки и тестирования	Z.500–Z.519
ПРОМЕЖУТОЧНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ	Z.600–Z.699
Среда распределенной обработки	Z.600–Z.609

Для получения более подробной информации просьба обращаться к перечню Рекомендаций МСЭ-Т.

Рекомендация МСЭ-Т Z.144

Нотация тестирования и управления тестом версии 3 (TTCN-3): Интерфейс времени выполнения (TRI)

Резюме

В настоящей Рекомендации приводится спецификация интерфейса времени выполнения для реализаций тестовой системы согласно TTCN-3 (*Нотация тестирования и управления тестом версии 3*). Интерфейс времени выполнения согласно TTCN-3 обеспечивает рекомендуемую адаптацию для синхронизации и связи тестовой системы соответственно с отдельной платформой обработки и с тестируемой системой. В настоящей Рекомендации данный интерфейс определяется как набор операций, независимых от целевого языка.

Этот интерфейс определяется как соответствующий положениям Рекомендации МСЭ-Т Z.140. Для полного описания интерфейса TRI в настоящей Рекомендации используется язык описания интерфейса (IDL) обобщенной архитектуры посредника объектных запросов (CORBA). Разделы 6 и 7 содержат описания отображений языка абстрактной спецификации на целевые языки Java и ANSI-C. В Приложении A содержится резюме, касающееся спецификации интерфейса на базе языка IDL.

Источник

Рекомендация МСЭ-Т Z.144 утверждена 16 марта 2006 года 17-й Исследовательской комиссией МСЭ-Т (2005–2008 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8.

ПРЕДИСЛОВИЕ

Международный союз электросвязи (МСЭ) является специализированным учреждением Организации Объединенных Наций в области электросвязи. Сектор стандартизации электросвязи МСЭ (МСЭ-Т) – постоянный орган МСЭ. МСЭ-Т отвечает за изучение технических, эксплуатационных и тарифных вопросов и за выпуск Рекомендаций по ним с целью стандартизации электросвязи на всемирной основе.

На Всемирной ассамблее по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяются темы для изучения Исследовательскими комиссиями МСЭ-Т, которые, в свою очередь, вырабатывают Рекомендации по этим темам.

Утверждение рекомендаций МСЭ-Т осуществляется в соответствии с процедурой, изложенной в Резолюции 1 ВАСЭ.

В некоторых областях информационных технологий, которые входят в компетенцию МСЭ-Т, необходимые стандарты разрабатываются на основе сотрудничества с ИСО и МЭК.

ПРИМЕЧАНИЕ

В настоящей Рекомендации термин "администрация" используется для краткости и обозначает как администрацию электросвязи, так и признанную эксплуатационную организацию.

Соблюдение положений данной Рекомендации носит добровольный характер. Однако в Рекомендации могут содержаться определенные обязательные положения (например, для обеспечения возможности взаимодействия или применимости), и соблюдение положений данной Рекомендации достигается в случае выполнения всех этих обязательных положений. Для выражения необходимости выполнения требований используется синтаксис долженствования и соответствующие слова (такие, как "должен" и т. п.), а также их отрицательные эквиваленты. Использование этих слов не предполагает, что соблюдение положений данной Рекомендации является обязательным для какой-либо из сторон.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на вероятность того, что практическое применение или реализация этой Рекомендации может включать использование заявленного права интеллектуальной собственности. МСЭ не занимает какую бы то ни было позицию относительно подтверждения, обоснованности или применимости заявленных прав интеллектуальной собственности, независимо от того, отстаиваются ли они членами МСЭ или другими сторонами вне процесса подготовки Рекомендации.

На момент утверждения настоящей Рекомендации МСЭ получил извещение об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для выполнения этой Рекомендации. Однако те, кто будет применять Рекомендацию, должны иметь в виду, что это может не отражать самую последнюю информацию, и поэтому им настоятельно рекомендуется обращаться к патентной базе данных БСЭ по адресу: http://www.itu.int/ITU-T/ipr/.

СОДЕРЖАНИЕ

Стр.

Введение

Настоящая Рекомендация состоит из двух отдельных частей. Первая часть содержит описание структуры реализации тестовой системы TTCN-3, а во второй части представлена спецификация интерфейса времени выполнения согласно TTCN-3.

В первой части представлены четыре основных объекта разбивки тестовой системы согласно TTCN-3, а именно:

- Администрирование тестом (TM);
- Выполняемый модуль TTCN-3 (TE);
- Адаптер системы SUT (SA); и
- Адаптер платформы (PA).

Кроме того, определены взаимодействия между этими объектами, т. е. соответствующие интерфейсы.

Во второй части настоящей Рекомендации дано описание интерфейса времени выполнения (TRI) согласно TTCN-3. Этот интерфейс определяется на основе операций, которые реализуются в виде части одного объекта и вызываются другими объектами тестовой системы. Для каждой операции в спецификации интерфейса определяются соответствующие структуры данных, ожидаемое влияние на тестовую систему и все ограничения на использование этой операции. Следует отметить, что в этой спецификации интерфейса определяются только взаимодействия между интерфейсом TSI и системой SUT, а также операции таймера.

Рекомендация МСЭ-Т Z.144

Нотация тестирования и управления тестом версии 3 (TTCN-3): Интерфейс времени выполнения (TRI)

1 Сфера применения

В настоящей Рекомендации приводится спецификация интерфейса времени выполнения для реализаций тестовой системы TTCN-3. Интерфейс времени выполнения согласно TTCN-3 обеспечивает стандартизированную адаптацию для синхронизации и связи тестовой системы соответственно с отдельной платформой обработки и с тестируемой системой. В настоящей Рекомендации данный интерфейс определяется как набор операций, независимых от целевого языка.

Этот интерфейс определяется как отвечающий стандарту TTCN-3 (см. ссылки ниже). Для полного описания интерфейса TRI в настоящей Рекомендации используется язык описания интерфейса (IDL) архитектуры CORBA. Разделы 6 и 7 содержат описание отображений языка данной абстрактной спецификации на целевые языки Java и ANSI-C. В Приложении A содержится резюме, касающееся спецификации интерфейса на базе языка IDL.

1.1 Соответствие

Требование соответствия тестовой системы TTCN-3 интерфейсу TRI состоит в следовании положениям спецификации интерфейса, изложенным в настоящей Рекомендации, а также одному из рассматриваемых отображений целевого языка.

ПРИМЕР: Если поставщик поддерживает язык Java, то вызовы и реализации операций интерфейса TRI, являющиеся частью выполняемого модуля TTCN-3, должны соответствовать отображению языка IDL на язык Java, описанному в данной Рекомендации.

2 Справочные документы

Указанные ниже Рекомендации МСЭ-Т и другие источники содержат положения, которые путем ссылки на них в данном тексте составляют положения настоящей Рекомендации. На момент публикации указанные издания были действующими. Все Рекомендации и другие источники могут подвергаться пересмотру; поэтому всем пользователям данной Рекомендации предлагается изучить возможность применения последнего издания Рекомендаций и других источников, перечисленных ниже. Список действующих в настоящее время Рекомендаций МСЭ-Т, регулярно публикуется. Ссылка на документ в данной Рекомендации не придает ему как отдельному документу статус Рекомендации.

[1] Рекомендация МСЭ-Т X.290 (1995), *Методология аттестационного тестирования ВОС и структура Рекомендаций о протоколах для применений МСЭ-Т – Общие понятия*.

ИСО/МЭК 9646-1:1994, *Информационные технологии – Взаимосвязь открытых систем – Методология и структура аттестационного тестирования – Часть 1: Общие понятия*.

[2] Рекомендация МСЭ-Т Z.140 (2006), *Нотация тестирования и управления тестом версии 3 (TTCN-3): Базовый язык*.

[3] Рекомендация МСЭ-Т X.292 (2002), *Методология аттестационного тестирования ВОС и структура Рекомендаций о протоколах для применений МСЭ-Т – Древовидно-табличная комбинированная нотация (TTCN)*.

ИСО/МЭК 9646-3:1998, *Информационные технологии – Взаимосвязь открытых систем – Методология и структура аттестационного тестирования – Часть 3: Древовидно-табличная комбинированная нотация (TTCN)*.

[4] Рекомендация МСЭ-Т Z.143 (2006), *Нотация тестирования и управления тестом версии 3 (TTCN-3): Операционная семантика*.

3 Определения и сокращения

3.1 Определения

В данной Рекомендации применяются термины и определения, приведенные в Рекомендации МСЭ-Т Z.140 [2], а также следующие определения:

3.1.1 абстрактная тестовая последовательность (abstract test suite) (ATS): См. Рекомендацию МСЭ-Т X.290 [1].

3.1.2 порт связи (communication port): Абстрактный механизм, облегчающий связь между тестовыми компонентами.

ПРИМЕЧАНИЕ. – Порт связи моделируется в виде очереди FIFO в направлении приема. Порты могут быть на базе сообщений, на базе процедур или на базе и того и другого.

3.1.3 выполняемая тестовая последовательность (executable test suite) (ETS): См. Рекомендацию МСЭ-Т X.290 [1].

3.1.4 явный таймер (explicit timer): Таймер, который объявлен в последовательности ATS TTCN-3 и доступ к которому может осуществляться через операции таймера TTCN-3.

3.1.5 дополнительная информация о реализации для тестирования (ДИРТ) (implementation extra information for testing) (IXIT): См. Рекомендацию МСЭ-Т X.290 [1].

3.1.6 неявный таймер (implicit timer): Системный таймер, создаваемый выполняемым модулем TTCN-3 для защиты вызова TTCN-3 или для выполнения операции.

ПРИМЕЧАНИЕ. – Неявные таймеры недоступны для пользователя TTCN-3.

3.1.7 адаптер платформы (platform adapter) (PA): Объект, который адаптирует выполняемый модуль TTCN-3 к отдельной платформе выполнения.

ПРИМЕЧАНИЕ. – Адаптер платформы создает единственное понятие времени для тестовой системы TTCN-3 и реализует внешние функции, а также явные и неявные таймеры.

3.1.8 адаптер SUT (SUT adapter) (SA): Объект, который адаптирует операции связи TTCN-3 с системой SUT на основе абстрактного интерфейса тестовой системы и осуществляет реальный интерфейс тестовой системы.

3.1.9 тестируемая система (system under test) (SUT): См. Рекомендацию МСЭ-Т X.290 [1].

ПРИМЕЧАНИЕ. – Во время компиляции известны все типы, т. е. они статически связаны.

3.1.10 тестовый пример (test case): См. Рекомендацию МСЭ-Т X.290 [1].

3.1.11 тестовое событие (test event): Принятые или переданные тестовые данные (вызов сообщения или процедуры) на порте связи, являющемся частью интерфейса тестовой системы.

3.1.12 администрирование тестом (test management) (TM): Объект, предоставляющий интерфейс пользователя и управляющий тестовой системой TTCN-3.

3.1.13 тестовая система (test system): См. Рекомендацию МСЭ-Т X.290 [1].

3.1.14 интерфейс тестовой системы (test system interface): Тестовый компонент, который обеспечивает отображение портов, доступных в (абстрактной) тестовой системе TTCN-3, в порты, предоставляемые реальной тестовой системой.

3.1.15 идентификация таймера (timer identification) (TID): Однозначно определяемая идентификация для экземпляров явного или неявного таймера, которая генерируется выполняемым модулем TTCN-3.

3.1.16 интерфейс управления TTCN-3 (TTCN-3 control interface) (TCI): В настоящее время это – частный интерфейс, определяющий взаимодействие между управлением тестом и выполняемым модулем TTCN-3 в тестовой системе.

3.1.17 выполняемый модуль TTCN-3 (TTCN-3 executable) (TE): Часть тестовой системы, которая отвечает за интерпретацию или выполнение тестовой последовательности ETS TTCN-3.

3.1.18 интерфейс времени выполнения TTCN-3 (TTCN-3 runtime interface) (TRI): Интерфейс, определяющий взаимодействие выполняемого модуля TTCN-3 с системой SUT и адаптером платформы в тестовой системе.

3.2 Сокращения

В настоящей Рекомендации используются следующие сокращения:

ATS	Abstract Test Suite	Абстрактная тестовая последовательность
CH	Component Handler	Обработчик компонентов
ECD	External CoDecs	Внешние кодеки
EDS	Encoding/Decoding System	Система кодирования/декодирования
ETS	Executable Test Suite	Выполняемая тестовая последовательность
IDL	Interface Definition Language	Язык описания интерфейса
IXIT	Implementation eXtra Information for Testing	Дополнительная информация о реализации для тестирования (ДИРТ)
MSC	Message Sequence Chart	Схема последовательности сообщений
MTC	Main Test Component	Главный тестовый компонент
OMG	Object Management Group	Рабочая группа по управлению объектами
PA	Platform Adapter	Адаптер платформы
SA	SUT Adapter	Адаптер системы SUT
SUT	System Under Test	Тестируемая система
T3RTS	TTCN-3 Runtime System	Система на этапе выполнения TTCN-3
TC	Test Control	Управление тестом
TCI	TTCN-3 Control Interface	Интерфейс управления TTCN-3
TE	TTCN-3 Executable	Выполняемый модуль TTCN-3
TID	Timer Identification	Идентификация таймера
TL	Test Logging	Тестовая регистрация
TM	Test Management	Администрирование тестом
TRI	TTCN-3 Runtime Interface	Интерфейс времени выполнения TTCN-3
TSI	Test System Interface	Интерфейс тестовой системы

TTCN	Testing and Test Control Notation	Нотация тестирования и управления тестом
TTCN-3	Tree and Tabular Combined Notation version 3	Древовидно-табличная комбинированная нотация версии 3

4 Общая структура тестовой системы TTCN-3

Тестовую систему TTCN-3 можно рассматривать концептуально в виде множества взаимодействующих объектов, где каждый объект соответствует отдельному аспекту функциональных возможностей при реализации тестовой системы. Эти объекты управляют выполнением тестов, интерпретацией или выполнением компилированных программ TTCN-3, реализуют необходимую связь с системой SUT, реализуют внешние функции и обрабатывают операции таймеров. (См. рисунок 1.)

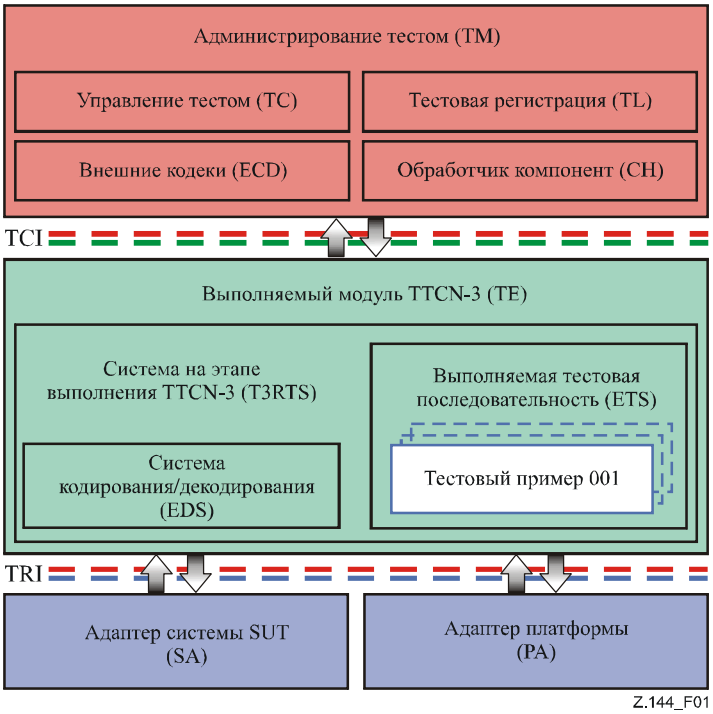

Рисунок 1/Z.144 – Общая структура тестовой системы TTCN-3

4.1 Объекты в тестовой системе TTCN-3

На рисунке 1 представлена структура реализации тестовой системы TTCN-3. Следует заметить, что дополнительная разбивка TM на более мелкие объекты, как показано на рисунке 1 и используемая в последующих разделах настоящей Рекомендации, служит исключительно для цели определения интерфейсов тестовой системы TTCN-3.

Часть тестовой системы, которая отвечает за интерпретацию и выполнение модулей TTCN-3, т. е. выполняемая тестовая последовательность (ETS), является частью выполняемого модуля TTCN-3 (TE). Это соответствует либо выполняемой программе, генерируемой компилятором TTCN-3, либо интерпретатору TTCN-3 в реализации тестовой системы. Предполагается, что в реализацию тестовой системы входит тестовая последовательность ETS, полученная из тестовой последовательности ATS TTCN-3.

Остальная часть тестовой системы TTCN-3, затрагивающая все аспекты, которые не могут быть получены из информации, имеющейся только в исходной ATS, может быть разложена на объекты: администрирование тестом (TM), адаптер системы SUT (SA) и адаптер платформы (PA). Вообще эти объекты охватывают интерфейс пользователя тестовой системы, управление выполнением тестов, регистрацию тестовых событий, а также связь с системой SUT и реализацией таймеров.

4.1.1 Администрирование тестом (TM)

В объекте TM можно делать различие между функциональными возможностями, связанными с управлением выполнения тестов, и регистрацией тестовых событий.

4.1.1.1 Управление тестом (TC)

Объект TC отвечает за общее администрирование тестовой системы. После инициализации тестовой системы выполнение тестов начинается в объекте TC. Этот объект отвечает за правильный вызов модулей TTCN-3, т. е. за передачу параметров модулей и/или информации IXIT на модуль TE, если это необходимо. Обычно этот объект также реализует интерфейс пользователя тестовой системы.

4.1.1.2 Тестовая регистрация (TL)

Объект TL отвечает за поддержку тестовой регистрации. Он явным образом уведомляет модуль TE о событиях регистрации тестов. Объект TL имеет однонаправленный интерфейс, по которому любая часть объекта модуля TE может послать объекту TL запрос регистрации. Внутренний интерфейс объекта TM может быть также использован для записи информации об администрировании тестом, генерируемой объектом TC.

4.1.1.3 Внешние кодеки (ECD)

Объекты внешних кодеков дополнительно отвечают за кодирование и декодирование данных, относящихся к связи на базе сообщений или процедур, в модуле TE. Эти внешние кодеки могут быть использованы параллельно со встроенными кодеками, связанными с модулем TE, или вместо них. В отличие от встроенных кодеков внешние кодеки имеют стандартизированный интерфейс, что позволяет перемещать их между разными системами и средствами TTCN-3.

4.1.1.4 Обработчик компонентов (CH)

Объект CH отвечает за распределение параллельных тестовых компонентов. Это распределение может осуществляться через одну или несколько физических систем. Объект CH дает возможность объекту администрирования тестом создавать распределенные тестовые системы и управлять ими прозрачным образом и независимо от модуля TE.

4.1.2 Выполняемый модуль TTCN-3 (TE)

Объект TE отвечает за интерпретацию либо выполнение последовательности ATS TTCN-3. Концептуально объект TE можно разбить на три взаимодействующих объекта: ETS, систему на этапе выполнения TTCN-3 (T3RTS) и систему кодирования/декодирования (EDS). Следует отметить, что такая разбивка объекта TE на более мелкие объекты концептуально служит только цели определения интерфейсов тестовой системы TTCN-3 – в реализациях интерфейса TRI нет требований для такой разбивки.

В последующих разделах определяются функции каждого объекта, а также приводится обсуждение обработки таймеров в интерфейсе TRI.

4.1.2.1 Выполняемая тестовая последовательность (ETS)

Объект ETS служит для обработки данных по выполнению или интерпретации тестовых примеров, по порядку следования и сопоставлению тестовых событий согласно описаниям в соответствующих модулях TTCN-3 исходя из Рекомендации МСЭ-Т Z.140 [2]. Этот объект взаимодействует с объектом T3RTS, чтобы передавать, пытаться принимать (или сопоставлять) и регистрировать тестовые события во время выполнения тестового примера, чтобы создавать и удалять тестовые компоненты TTCN-3, а также обрабатывать обращения к внешним функциям, команды действия и таймеры. Следует заметить, что объект ETS непосредственно не взаимодействует через интерфейс TRI с адаптером SA.

4.1.2.2 Система на этапе выполнения TTCN-3 (T3RTS)

Объект T3RTS взаимодействует с объектами TM, SA и PA через интерфейсы TCI и TRI и осуществляет администрирование объектами ETS и EDS. Объект T3RTS инициализирует адаптеры, а также объекты ETS и EDS. Этот объект выполняет все действия, необходимые для правильного запуска выполнения тестового примера или функции с параметрами в объекте ETS. Он запрашивает у объекта TM значения параметров модуля, требуемые для ETS, и посылает ему информацию регистрации. Он также собирает данные и выносит соответствующие вердикты, возвращаемые объектом ETS, как это определено в Рекомендации МСЭ-Т Z.140 [2].

Объект T3RTS осуществляет создание и удаление тестовых компонентов, а также семантики TTCN-3 связи на базе сообщений и процедур, обращений к внешним функциям, команд действий и таймеров. Сюда входит уведомление адаптера SUT (SA) о том, какой вызов сообщения или процедуры должен быть передан системе SUT, или же уведомление для адаптера платформы (PA) о том, какая внешняя функция должна быть выполнена или какие таймеры должны быть запущены, остановлены, запрошены или считаны. Подобным образом, объект T3RTS уведомляет объект ETS о входящих вызовах или обращениях от системы SUT к процедурам, а также о событиях, связанных с тайм-аутом.

Объект T3RTS активизирует объект EDS перед передачей или приемом сообщений и перед обращениями к процедурам для адаптера SA или от него, или перед обработкой вызовов функций и команд действий в адаптере PA для объекта ETS с целью их кодирования или декодирования. Объект T3RTS должен осуществлять все операции со связью между тестовыми компонентами на базе сообщений и процедур, но только с соблюдением семантики TTCN-3 связи на базе процедур с системой SUT, т. е. при возможной блокировке и разблокировке выполнения тестового компонента, защите с неявными таймерами и обработке особых состояний по тайм-аутам в результате таких операций связи. В адаптере SA должны быть реализованы и определены (в случае операции приема) все операции связи на базе процедур с системой SUT, поскольку они

наиболее эффективно реализуются характерным для платформы образом. Следует отметить, что синхронизация любой операции с обращением к процедуре, т. е. к неявным таймерам, реализуется в адаптере платформы (PA).

Чтобы выполнить отображения мгновенного состояния процесса для операций приема согласно Рекомендации МСЭ-Т Z.140 [2] необходимо, чтобы выполняемый модуль TTCN-3 поддерживал очереди на своем собственном порте (отличаются от очередей, которые могут быть доступными в адаптерах SA или PA). События срабатывания таймера, генерируемые таймером TTCN-3, таймером вызова или таймером тестового примера, должны сохраняться в списке тайм-аутов согласно Рекомендации МСЭ-Т X.292 [3]. На рисунке 2 все эти функциональные возможности присвоены объекту T3RTS. Он отвечает за хранение событий, о которых модуль TE уведомил адаптеры SA или PA, но которые еще должны быть обработаны.

4.1.2.3 Система кодирования/декодирования (EDS)

Объект EDS отвечает за кодирование и декодирование тестовых данных, в которые входят данные, используемые в операциях связи с системой SUT, как определено в исполнительном модуле TTCN-3. Если для модуля TTCN-3 кодирование не задано, то кодирование значений данных определяется инструментальными средствами. Этот объект активизируется объектом T3RTS с передачей результатов последнему. Следует отметить, что объект EDS непосредственно не взаимодействует через интерфейс TRI с адаптером SA.

4.1.2.4 Таймеры в выполняемом модуле TTCN-3

Таймеры, объявленные и именованные в тестовой последовательности ATS TTCN-3, могут быть концептуально классифицированы в модуле TE как явные. Таймеры, созданные в модуле TE для защиты обращений к процедурам TTCN-3 или исполнительных операций, известны в модуле TE в качестве неявных таймеров. Как явные, так и неявные таймеры создаются в модуле TE, но реализуются адаптером платформы (PA). Это достигается путем генерирования однозначно определяемой идентификации таймера (TID) для любого таймера, созданного в модуле TE. Эта однозначно определяемая идентификация TID должна позволять модулю TE отличать разные таймеры друг от друга. Идентификация TID должна использоваться модулем TE для взаимодействия с соответствующей реализацией таймера в адаптере PA.

Следует отметить, что модуль TE отвечает за правильную реализацию разной семантики TTCN-3 для явных и неявных таймеров согласно Рекомендации МСЭ-Т Z.140 [2], например, использование ключевых слов `any` и `all` таймерами применимо только к явным таймерам. Все таймеры в адаптере PA, т. е. явные и неявные, рассматриваются таким же образом.

4.1.3 Адаптер системы SUT (SA)

Адаптер SA приспосабливает связь с системой SUT на базе сообщений и процедур тестовой системы TTCN-3 к отдельной исполнительной платформе тестовой системы. Он осведомлен об отображении портов связи тестового компонента TTCN-3 на порты интерфейсов тестовой системы и осуществляет реальный интерфейс тестовой системы согласно Рекомендации МСЭ-Т Z.140 [2]. Он отвечает за передачу запросов на передачу и команд действия системы SUT от выполняемого модуля TTCN-3 (TE) к системе SUT и за уведомление модуля TE о всех принятых тестовых событиях путем присоединения их к очередям в портах модуля TE.

В адаптере SA реализуются все операции связи с системой SUT на базе процедур. Адаптер SA отвечает за установление различий между разными сообщениями в пределах связи на базе процедур (т. е. вызов, ответ и особое состояние) и за передачу их соответствующим образом либо к системе SUT, либо к модулю TE. В модуле TE должна обрабатываться семантика связи на базе процедур TTCN-3, т. е. влияние такой операции на выполнение тестового компонента TTCN-3.

Адаптер SA имеет интерфейс с модулем TE, который используется для передачи сообщений системы SUT (выдаваемых в командах действий системы SUT TTCN-3) к адаптеру SA и для обмена закодированными тестовыми данными между двумя объектами в операциях связи с системой SUT.

4.1.4 Адаптер платформы (PA)

Адаптер PA реализует внешние функции TTCN-3 и обеспечивает тестовую систему TTCN-3 с единственным представлением времени. В этом объекте должны быть реализованы внешние функции, а также все таймеры. Следует отметить, что экземпляры таймеров создаются в модуле TE. Таймер в адаптере PA можно отличить только по его идентификации таймера (TID). Поэтому в адаптере PA как явные, так и неявные таймеры обрабатываются по одному и тому же способу.

Интерфейс с модулем TE позволяет активизировать внешние функции и запуск, чтение и останов таймеров, а также осуществлять запрос о состоянии таймеров, используя их идентификацию ID таймера. Адаптер PA уведомляет модуль TE о таймерах с истекшим временем действия.

4.2 Интерфейсы в тестовой системе TTCN-3

Как было показано на рисунке 1, тестовая система TTCN-3 имеет два интерфейса, а именно, интерфейс управления TTCN-3 (TCI) и интерфейс времени выполнения TTCN-3 (TRI), который задает интерфейс соответственно между администрированием тестом (TM) и выполняемым модулем и между объектами TE, адаптером системы SUT (SA) и адаптером платформы (PA).

В данной Рекомендации определяется интерфейс TRI. Взаимодействие модуля TE с адаптерами SA и PA будет определяться здесь в терминах операций интерфейса TRI. И хотя оба интерфейса, т. е. TRI и TCI, должны быть определены для полной реализации тестовой системы TTCN-3, спецификация и реализация интерфейса TCI в настоящее время считаются частной собственностью.

4.3 Требования к выполнению для тестовой системы TTCN-3

Каждый вызов операции интерфейса TRI будет интерпретироваться в качестве элементарной операции в вызывающем объекте. Вызываемый объект, который выполняет операцию интерфейса TRI, будет возвращать управление вызывающему объекту, как только будет получен ожидаемый им результат, или если операция не может быть успешно завершена. Вызываемый объект не должен препятствовать реализации связи на базе процедур. Тем не менее вызываемый объект будет выполнять блокирование после инициирования реализации внешней функции и ожидать возвращаемого этой функцией значения. Следует заметить, что в зависимости от реализации тестовой системы ошибки в возвращаемом значении при реализации внешней функции могут привести к постоянной блокировке выполнения тестового компонента, выполняемого модуля TTCN-3, адаптера платформы и даже всей тестовой системы.

Сформулированные выше требования к выполнению могут быть осуществлены при реализации хорошо интегрированной тестовой системы. Здесь вся тестовая система TTCN-3 реализуется в одном выполняемом модуле или процессе, где каждому объекту тестовой системы выделяется по меньшей мере одна цепочка выполнения тестирования. Здесь операции интерфейса TRI могут быть реализованы как вызовы процедур.

Следует отметить, что все же возможно ухудшение интеграции при реализации тестовой системы, например при реализации тестовой системы TTCN-3 с множеством адаптеров системы SUT в распределенной вычислительной среде. В этом случае только малая часть адаптера системы SUT хорошо интегрирована с остальной частью тестовой системы TTCN-3, тогда как фактические адаптеры SA могут быть реализованы в отдельных процессах. Эта малая часть адаптера SA может затем осуществлять только маршрутизацию информации, предоставляемой с помощью операций интерфейса TRI для необходимых процессов адаптера системы SUT, возможно, обрабатываемой в удаленных хост-узлах, и наоборот.

5 Интерфейс времени выполнения TTCN-3 и операции

Операции интерфейса TRI в данном разделе определяются исходя из того, когда они должны быть использованы и каков ожидаемый результат от их выполнения при реализации тестовой системы TTCN-3. Также устанавливается множество абстрактных типов данных, которое затем используется для определения операций интерфейса TRI. В это определение также входит более подробное описание входных параметров, необходимых для каждого вызова операции интерфейса TRI и ее возвращаемого значения.

5.1 Обзор интерфейса TRI

Интерфейс TRI определяет взаимодействие между объектами: выполняемым модулем TTCN-3 (TE), адаптером системы SUT (SA) и адаптером платформы (PA) в пределах реализации тестовой системы TTCN-3. Концептуально он предоставляет средства модулю TE для передачи тестовых данных системе SUT или для манипулирования таймерами и подобным образом для уведомления модуля TE о полученных тестовых данных и тайм-аутах.

Можно считать, что интерфейс TRI состоит из двух субинтерфейсов: интерфейса triCommunication и интерфейса triPlatform. Интерфейс triCommunication отвечает за связь последовательности ETS TTCN-3 с системой SUT, реализуемую в адаптере SA. Интерфейс triPlatform представляет множество операций, которые приспосабливают последовательность ETS к отдельной платформе выполнения.

Оба интерфейса являются двунаправленными, так что вызывающая и вызываемые стороны располагаются в объектах TE, SA и PA тестовой системы. В таблице 1 более подробно показана взаимосвязь "вызывающий/вызываемый" между соответствующими объектами. Следует заметить, что в этой таблице представлены только взаимодействия, видимые в интерфейсе TRI. Внутренняя связь между частями одного и того же объекта не отражена, поскольку внутренняя структура объектов TE, SA или PA может отличаться при реализации тестовой системы TTCN-3.

Таблица 1/Z.144 – Обзор интерфейса

Интерфейс	Направление (вызывающий объект → вызываемый объект)	
Имя	TE → SA или PA	SA или PA → TE
triCommunication	TE → SA	SA → TE
triPlatform	TE → PA	PA → TE

5.1.1 Интерфейс triCommunication

Этот интерфейс состоит из операций, необходимых для реализации связи последовательности ETS TTCN-3 с системой SUT. Он включает операции для инициализации интерфейса тестовой системы (TSI), установления соединений к системе SUT и обработки связи с системой SUT на базе сообщений и процедур. Кроме того, в интерфейсе triCommunication предлагается операция по сбросу адаптера системы SUT (SA).

5.1.2 Интерфейс triPlatform

Этот интерфейс содержит все операции, необходимые для приспособления выполняемого модуля TTCN-3 к отдельной платформе выполнения. Интерфейс triPlatform предоставляет средства для запуска, останова, считывания таймера, запроса о его состоянии и для ввода событий, связанных с тайм-аутом, в список таймеров, с истекшим временем действия. Кроме того, он предоставляет операции по вызову внешних функций TTCN-3 и по сбросу адаптера платформы (PA). Следует заметить, что нет разницы между явными и неявными таймерами, необходимыми для интерфейса triPlatform. Просто адресация каждого таймера будет однозначно определяемой с помощью его идентификатора таймера (TID).

5.1.3 Корреляция между инициированиями операций TTCN-3 и TRI

Как показано в таблице 2, существует положительная корреляция между некоторыми инициированиями операций TTCN-3 и одним инициированием операции интерфейса TRI (или, возможно, двумя инициированиями в случае операций `execute` и `call` TTCN-3). Для всех других инициирований операции интерфейса TRI положительной корреляции может не существовать.

Показанная в таблице корреляция для операций связи TTCN-3 (т. е. для операций `send`, `call`, `reply` и `raise`) сохраняется, если только эти операции инициируются на порте тестового компонента, который отображается на порт интерфейса TSI. Тем не менее эта корреляция сохраняется для всех инициирований таких операций, если для тестового примера не был задан системный компонент, т. е. если для тестового примера создан только тестовый компонент MTC и никакие другие тестовые компоненты.

Таблица 2/Z.144 – Корреляция между инициированиями операций TTCN-3 и TRI (* = если применяется)

Имя операции TTCN-3	Имя операции TRI	Имя операции TRI
`execute`	`triExecuteTestCase` `triStartTimer*`	`TriCommunication` `TriPlatform`
`map`	`triMap`	`TriCommunication`
`unmap`	`triUnmap`	`TriCommunication`
`send`	`triSend (см. примечание 1)`	`TriCommunication`
	`triSendBC (см. примечание 2)`	
	`triSendMC (см. примечание 3)`	
`call`	`triCall (см. примечание 1)`	`TriCommunication`
	`triCallBC (см. примечание 2)`	
	`triCallMC (см. примечание 3)`	
	`triStartTimer*`	`TriPlatform`
`reply`	`triReply (см. примечание 1)`	`TriCommunication`
	`triReply (см. примечание 2)`	
	`triReply (см. примечание 3)`	
`raise`	`triRaise (см. примечание 1)`	`TriCommunication`
	`triRaise (см. примечание 2)`	
	`triRaise (см. примечание 3)`	
`action`	`triSUTactionInformal`	`TriCommunication`
`start (timer)`	`triStartTimer`	`TriPlatform`
`stop (timer)`	`triStopTimer`	`TriPlatform`
`read (timer)`	`triReadTimer`	`TriPlatform`
`running (timer)`	`triTimerRunning`	`TriPlatform`
`TTCN-3 external function`	`triExternalFunction`	`TriPlatform`
ПРИМЕЧАНИЕ 1. – Для одноадресной связи. ПРИМЕЧАНИЕ 2. – Для широковещательной связи. ПРИМЕЧАНИЕ 3. – Для многоадресной связи.		

Следует заметить, что все операции TRI, перечисленные в таблице 2, используются модулем TE и что модуль TE может выполнять инициирование этих операций по-разному при оценке отображения мгновенного состояния TTCN в рамках последовательности ETS TTCN-3.

5.2 Обработка ошибок

Обработка явных ошибок предусматривается только для операций интерфейса TRI, вызываемых выполняемым модулем TTCN-3 (TE). Адаптеры SA или PA сообщают о состоянии операции интерфейса TRI в возвращаемом значении операции TRI. Значение состояния может указывать либо на локальный успех (***TRI_OK***), либо на ошибку (***TRI_Error***) операции интерфейса TRI. Поэтому модуль TE может реагировать на ошибку, которая имела место либо в SA, либо в PA, и выдавать, например, информацию об ошибке в тестовом примере.

Для операций, вызываемых адаптерами SA или PA, обработка явных ошибок не требуется, поскольку эти операции выполняются в модуле TE. Здесь модуль TE управляет выполнением теста, в случае если эта ошибка имеет место в такой операции интерфейса TRI.

Следует заметить, что вопросы рассмотрения кодов характерных ошибок, а также обнаружения и обработки ошибок в любом из объектов тестовой системы выходят за рамки настоящей спецификации интерфейса TRI.

5.3 Интерфейс данных

В операциях TRI должны пересылаться только закодированные тестовые данные. Выполняемый модуль TTCN-3 (TE) отвечает за кодирование тестовых данных, подлежащих передаче, и за декодирование принимаемых тестовых данных в соответствующих операциях TRI, поскольку правила кодирования могут быть заданы для модуля TTCN-3 или в самом модуле. Следует заметить, что требуется, чтобы модуль TE кодировал тестовые данные, даже если информация кодирования не была предоставлена в последовательности ATS TTCN-3. В этом случае поставщик инструментальных средств должен описать кодирование.

Вместо определения явного интерфейса данных для TTCN-3 и типов данных ASN.1, в стандарте TRI определяется множество абстрактных типов данных. Эти типы данных используются в последующем определении операций TRI для указания на то, какая информация должна пересылаться от вызывающего объекта к вызываемому и наоборот. Конкретное представление этих абстрактных типов данных, а также определение базовых типов данных заданы в соответствующих языковых отображениях в разделах 6 и 7.

Следует заметить, что значения для любого типа данных идентификатора будут уникальными в реализации тестовой системы, где уникальность определяется как глобально различающаяся в любой временно́й точке.

Для определения операций интерфейса TRI установлены и используются следующие абстрактные типы данных.

5.3.1 Соединение

`TriComponentIdType` Значение типа `TriComponentIdType` содержит идентификатор, имя и тип компонента. Особое значение последнего – это имя типа компонента, как задано в последовательности ATS TTCN-3. Этот абстрактный тип главным образом используется для разрешения операций связи TRI в портах TSI, имеющих отображения на несколько портов тестовых компонентов.

`TriComponentIdListType` Значение типа `TriComponentIdListType` – это список типа `TriComponentIdType`. Этот абстрактный тип используется для многоадресной связи в TCI.

`TriPortIdType` Значение типа `TriPortIdType` включает значение типа `TriComponentIdType` для представления компонента, к которому принадлежит порт, индекс порта (если имеется) и имя порта, как указано в последовательности ATS TTCN-3. Тип `TriPortIdType` требуется главным образом для пересылки информации о TSI и соединениях с TSI от TE к SA.

`TriPortIdListType` Значение типа `TriPortIdListType` – это список типа `TriPortIdType`. Этот абстрактный тип используется в целях инициализации после инициирования тестового примера TTCN-3.

5.3.2 Связь

`TriMessageType` Значение типа `TriMessageType` – это закодированные тестовые данные, которые либо должны быть посланы к SUT, либо получены от SUT.

`TriAddressType` Значение типа `TriAddressType` указывает на адрес источника или места назначения в SUT. Этот абстрактный тип может быть использован в операциях связи TRI; он является открытым типом, который непрозрачен для TE.

`TriAddressListType` Значение типа `TriAddressListType` – это список типа `TriAddressType`. Этот абстрактный тип используется для многоадресной связи в TRI.

`TriSignatureIdType` Значение типа `TriSignatureIdType` – это имя сигнатуры процедуры, как описано в последовательности ATS TTCN-3. Этот абстрактный тип используется в операциях связи TRI на базе процедур.

`TriParameterType` Значение типа `TriParameterType` включает закодированный параметр и значение типа `TriParameterPassingModeType` для представления режима пересылки, заданного для параметра в последовательности ATS TTCN-3.

`TriParameterPassingModeType` Значение типа `TriParameterPassingModeType` – это либо *in*, *inout*, либо *out*. Этот абстрактный тип используется в операциях связи TRI на базе процедур и для внешних функциональных вызовов.

`TriParameterListType` Значение типа `TriParameterListType` – это список типа `TriParameterType`. Этот абстрактный тип используется в операциях связи TRI, на базе процедур и для внешних функциональных вызовов.

`TriExceptionType` Значение типа `TriExceptionType` – это закодированный тип и значение особого состояния, которое должно быть послано к SUT или было принято от SUT. Этот абстрактный тип используется в операциях связи TRI на базе процедур.

5.3.3 Таймер

`TriTimerIdType` — Значение типа `TriTimerIdType` описывает идентификатор для таймера. Этот абстрактный тип требуется для всех операций таймера интерфейса TRI.

`TriTimerDurationType` — Значение типа `TriTimerDurationType` определяет период времени в секундах для таймера.

5.3.4 Разное

`TriTestCaseIdType` — Значение типа `TriTestCaseIdType` является именем тестового примера, как указано в последовательности ATS TTCN-3.

`TriFunctionIdType` — Значение типа `TriFunctionIdType` – это имя внешней функции, как указано в последовательности ATS TTCN-3.

`TriStatusType` — Значение типа `TriStatusType` – это либо ***TRI_OK***, либо ***TRI_Error***, указывающие на успех или ошибку операции интерфейса TRI.

5.4 Описание операций

Все описания операций выполняются с использованием языка описания интерфейса (IDL). Отображения конкретных языков определяются в разделах 6 и 7.

Все параметры *in*, *inout* и *out*, перечисленные в отдельном определении операции, являются обязательными для каждого вызова операции интерфейса TRI. Значение параметра *in* указывается вызывающим объектом. Подобным образом значение параметра *out* указывается вызываемым объектом. В случае параметра *inout* значение сначала указывается вызывающим объектом, но может быть заменено новым значением со стороны вызываемого объекта. Следует заметить, что хотя в TTCN-3 для определений сигнатур также используются параметры *in*, *inout* и *out*, обозначения, используемые в спецификации языка IDL интерфейса TRI, не связаны с обозначениями в спецификации TTCN-3.

В вызовах операций должно использоваться резервное значение, указывающее на отсутствие параметров, которые определены в соответствующем описании параметров интерфейса TRI в качестве факультативных. Резервные значения для этих типов определяются в каждом отображении языка, и на них будут затем ссылаться как на значение `null`.

Все функции в интерфейсе описываются с использованием следующего шаблона:

Функции	**Имя операции** вызывающий объект → вызываемый объект
Сигнатура	IDL-сигнатура.
Параметры In	Описание данных, пересылаемых в качестве параметров к операции от вызывающего объекта к вызываемому объекту.
Параметры Out	Описание данных, пересылаемых в качестве параметров к операции от вызываемого объекта к вызывающему объекту.
Параметры InOut	Описание данных, пересылаемых в качестве параметров к операции от вызывающего объекта к вызываемому объекту и обратно от вызываемого объекта к вызывающему объекту.
Возвращаемое значение	Описание данных, возвращаемых от операции к вызывающему объекту.
Ограничения	Описание всех ограничений, которые применяются к вызову операции.
Результат	Поведение, требуемое от вызываемого объекта до того, как можно будет возвратить данные об операции.

5.5 Операции интерфейса связи

5.5.1 Операция triSAReset (TE → SA)

Сигнатура	`TriStatusType triSAReset()`
Параметры In	не применяется
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triSAReset`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция может быть вызвана модулем TE для сброса адаптера SA в любое время.
Результат	Адаптер SA будет сбрасывать все средства связи, которые он поддерживает, например сброс статических соединений с системой SUT, закрытие динамических соединений с SUT, сброс всех ждущих обработки сообщений или вызовов процедур. Операция `triResetSA` в случае успешного выполнения операции возвращает ***TRI_OK***, в противном случае возвращает ***TRI_Error***.

5.5.2 Операции обработки соединений

5.5.2.1 Операция triExecuteTestCase (TE → SA)

Сигнатура	`TriStatusType triExecuteTestCase( in TriTestCaseIdType testCaseId, in TriPortIdListType tsiPortList)`
Параметры In	`testCaseId` идентификатор тестового примера для возможного выполнения `tsiPortList` список портов интерфейса тестовой системы, определяемых для тестовой системы
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triExecuteTestCase`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error*)** операции.
Ограничения	Эта операция вызывается модулем TE немедленно до выполнения любого тестового примера. На тестовый пример, который может быть выполнен, указывает `testCaseId`. `tsiPortList` содержит все порты, которые были объявлены в определении компонента системы для тестового примера, т. е. порты интерфейса TSI. Если компонент системы не был явно определен для тестового примера в последовательности ATS TTCN-3, тогда список `tsiPortList` содержит все порты связи тестового компонента MTC. Порты в списке `tsiPortList` упорядочены так, как они представлены в соответствующем описании компонента TTCN-3.
Результат	Адаптер SA может устанавливать все статические соединения с SUT и инициализировать для портов интерфейса TSI все средства связи. Операция `triExecuteTestCase` в случае успешного завершения операции возвращает ***TRI_OK***; в противном случае возвращает ***TRI_Error***.

5.5.2.2 Операция triMap (TE → SA)

Сигнатура	`TriStatusType triMap( in TriPortIdType compPortId, in TriPortIdType tsiPortId)`
Параметры In	`compPortId` идентификатор порта тестового компонента, подлежащего отображению `tsiPortId` идентификатор порта интерфейса тестовой системы, подлежащего отображению
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triMap`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию отображения TTCN-3.
Результат	Адаптер SA может устанавливать динамическое соединение с системой SUT для порта TSI, на который сделана ссылка. Операция `triMap` возвращает ***TRI_Error*** в случае, если соединение не могло быть успешно установлено; в противном случае возвращает ***TRI_OK***. Операция должна возвращать ***TRI_OK***, когда от тестовой системы не требуется устанавливать динамическое соединение.

5.5.2.3 Операция triUnmap (TE → SA)

Сигнатура	`TriStatusType triUnmap( in TriPortIdType compPortId,` `in TriPortIdType tsiPortId)`
Параметры In	`compPortId` идентификатор порта тестового компонента, подлежащего отображению `tsiPortId` идентификатор порта интерфейса тестовой системы, подлежащего отображению
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triUnmap`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет любую операцию неотображения TTCN-3.
Результат	Адаптер SA будет замыкать цепь динамического соединения с системой SUT для порта TSI, на который сделана ссылка. Операция `triUnmap` возвращает ***TRI_Error*** в случае, когда цепь соединения не может быть успешно замкнута или когда такое соединение ранее не было установлено, в противном случае возвращает ***TRI_OK***. В случае, когда тестовой системой не должны устанавливаться динамические соединения, операция должна возвращать ***TRI_OK***.

5.5.3 Операции связи на базе сообщений

5.5.3.1 Операция triSend (TE → SA)

Сигнатура	`TriStatusType triSend( in TriComponentIdType componentId,`
	`in TriPortIdType tsiPortId,`
	`in TriAddressType SUTaddress,`
	`in TriMessageType sendMessage)`

Параметры In	`componentId` идентификатор передающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который сообщение посылается адаптеру системы SUT `SUTaddress` адрес (факультативный) места назначения в системе SUT `sendMessage` закодированное сообщение для передачи
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triSend`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию многоадресной передачи TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций передачи TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Кодирование операции `sendMessage` должно осуществляться в модуле TE до этого вызова операции интерфейса TRI.
Результат	Адаптер SA может посылать сообщение к системе SUT. Операция `triSend` возвращает ***TRI_OK*** в случае, если она была завершена успешно. В противном случае будет возвращено значение ***TRI_Error***. Следует заметить, что возвращаемое значение ***TRI_OK*** не означает, что система SUT получила сообщение `sendMessage`.

5.5.3.2 Операция triSendBC (TE → SA)

Сигнатура	`TriStatusType triSendBC( in TriComponentIdType componentId,`
	`in TriPortIdType tsiPortId,`
	`in TriMessageType sendMessage)`

Параметры In	`componentId` идентификатор передающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который сообщение посылается адаптеру системы SUT `sendMessage` закодированное сообщение для передачи
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triSend`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию широковещательной передачи TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций передачи TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создается только тестовый компонент MTC. Кодирование сообщения `sendMessage` должно осуществляться в модуле TE до данного вызова операции интерфейса TRI.
Результат	Адаптер SA может выполнить широковещательную передачу сообщения к системе SUT. Операция `triSend` возвращает ***TRI_OK*** в случае, если она была завершена успешно. В противном случае будет возвращено значение ***TRI_Error***. Следует заметить, что возвращаемое значение ***TRI_OK*** не означает, что система SUT получила сообщение `sendMessage`.

5.5.3.3 Операция triSendMC (TE → SA)

Сигнатура	`TriStatusType triSendMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriMessageType sendMessage)`
Параметры In	`componentId` идентификатор передающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который сообщение посылается адаптеру системы SUT `SUTaddresses` адреса мест назначения в системе SUT `sendMessage` закодированное сообщение, подлежащее передаче
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triSend`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию многоадресной передачи TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций передачи TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Кодирование сообщения `sendMessage` должно осуществляться в модуле TE до этого вызова операции интерфейса TRI.
Результат	Адаптер SA может осуществлять многоадресную передачу сообщения к системе SUT. Операция `triSend` возвращает значение ***TRI_OK*** в случае, если она была завершена успешно. В противном случае будет возвращено значение ***TRI_Error***. Следует заметить, что возвращаемое значение ***TRI_OK*** не означает, что система SUT получила сообщение `sendMessage`.

5.5.3.4 Операция triEnqueueMsg (SA → TE)

Сигнатура	`void triEnqueueMsg( in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriComponentIdType componentId, in TriMessageType receivedMessage)`
Параметры In	`tsiPortId` идентификатор порта интерфейса тестовой системы, через который сообщение ставится в очередь адаптером системы SUT `SUTaddress` (факультативный) адрес источника в системе SUT `componentId` идентификатор принимающего тестового компонента `receivedMessage` закодированное принятое сообщение
Параметры Out	не применяется
Возвращаемое значение	пустое значение
Ограничения	Эта операция вызывается адаптером SA после того, как он получил сообщение от системы SUT. Она может использоваться только тогда, когда идентификатор tsiPortId либо был раньше отображен на порт с идентификатором компонента, либо на него была ссылка в предшествующем операторе triExecuteTestCase. При инициировании операции `triEnqueueMsg` сообщение `receivedMessage` будет содержать закодированное значение.
Результат	Эта операция будет пересылать сообщение к модулю TE, указывая на идентификатор componentId компонента, на который отображен идентификатор tsiPortId порта интерфейса TSI. Декодирование сообщения receivedMessage должно осуществляться в модуле TE.

5.5.4 Операции связи на базе процедур

5.5.4.1 Операция triCall (TE → SA)

Сигнатура	`TriStatusType triCall( in TriComponentIdType componentId,`
	`in TriPortIdType tsiPortId,`
	`in TriAddressType SUTaddress,`
	`in TriSignatureIdType signatureId,`
	`in TriParameterListType parameterList)`

Параметры In	`componentId` идентификатор тестового компонента, формирующего вызов процедуры `tsiPortId` идентификатор порта интерфейса тестовой системы, через который вызов процедуры передается адаптеру системы SUT `SUTaddress` адрес (факультативный) места назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triCall`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию одноадресного вызова TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций вызова TTCN-3, если для тестового примера не был задан компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Все параметры процедур *in* и *inout* содержат закодированные значения. Параметры процедуры – это параметры, заданные в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI.
Результат	При инициировании этой операции адаптер SA может инициировать вызов процедуры, соответствующий идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triCall` будет возвращать значение, не ожидая возврата значения сформированного вызова процедуры (см. примечание). Эта операция интерфейса TRI возвращает значение *TRI_OK* при успешном инициировании вызова процедуры; в противном случае она возвращает значение *TRI_Error*. В том случае, когда значение любого параметра *out* является ненулевым, адаптер SA указывает на отсутствие ошибки. Следует отметить, что возвращаемое значение этой операции интерфейса TRI не приводит к выводу об успехе или ошибке вызова процедуры. Следует отметить, что факультативное значение тайм-аута, которое может быть задано в последовательности ATS TTCN-3 для операции вызова, *не* включается в сигнатуру операции `triCall`. Модуль TE должен адресовать это формирование вызова путем запуска таймера для операции вызова TTCN-3 в адаптере PA с отдельным вызовом операции интерфейса TRI, т. е. `triStartTimer`.
ПРИМЕЧАНИЕ. – Этого можно достичь, например, путем порождения нового потока или процесса. Однако данная обработка этого вызова процедуры зависит от реализации модуля TE.	

5.5.4.2 Операция triCallBC (TE → SA)

Сигнатура	`TriStatusType triCallBC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriSignatureIdType signatureId, in TriParameterListType parameterList)`
Параметры In	`componentId` идентификатор тестового компонента, формирующего вызов процедуры `tsiPortId` идентификатор порта интерфейса тестовой системы, через который вызов процедуры передается адаптеру системы SUT `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triCall`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию широковещательного вызова TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций вызова TTCN-3, если для тестового примера не был описан компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Все параметры процедуры *in* и *inout* содержат закодированные значения. Параметры процедуры – это параметры, заданные в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI.
Результат	При инициировании этой операции адаптер SA может инициировать и осуществлять широковещательную передачу вызова процедуры, соответствующего идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triCall` будет возвращать значение, не ожидая возврата значения сформированного вызова процедуры (см. примечание). Эта операция интерфейса TRI возвращает значение *TRI_OK* при успешном инициировании вызова процедуры; в противном случае она возвращает значение *TRI_Error*. В том случае, когда значение любого параметра *out* является ненулевым, адаптер SA указывает на отсутствие ошибки. Следует отметить, что возвращаемое значение этой операции интерфейса TRI не приводит к выводу об успехе или ошибке вызова процедуры. Следует отметить, что факультативное значение тайм-аута, которое может быть задано в последовательности ATS TTCN-3 для операции вызова, *не* включено в сигнатуру операции `triCall`. Модуль TE должен адресовать это формирование вызова путем запуска таймера для операции вызова TTCN-3 в адаптере PA с отдельным вызовом операции интерфейса TRI, т. е. `triStartTimer`.
ПРИМЕЧАНИЕ. – Этого можно было бы достичь, например, путем порождения нового потока или процесса. Однако данная обработка этого вызова процедуры зависит от реализации модуля TE.	

5.5.4.3 Операция triCallMC (TE → SA)

Сигнатура	`TriStatusType triCallMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriParameterListType parameterList)`
Параметры In	`componentId` идентификатор тестового компонента, формирующего вызов процедуры `tsiPortId` идентификатор порта интерфейса тестовой системы, через который вызов процедуры передается адаптеру системы SUT `SUTaddresses` адреса мест назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triCall`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию многоадресного вызова TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций вызова TTCN-3, если для тестового примера не был задан компонент системы, т. е. только тестовый компонент MTC создается для тестового примера. Все параметры процедуры *in* и *inout* содержат закодированные значения. Параметры процедуры – это параметры, заданные в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI.
Результат	При инициировании этой операции адаптер SA может инициировать и осуществлять многоадресную передачу вызова процедуры, соответствующего идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triCall` будет возвращать значение, не ожидая возврата значения сформированного вызова процедуры (см. примечание). Эта операция интерфейса TRI возвращает значение *TRI_OK* при успешном инициировании вызова процедуры; в противном случае возвращается значение *TRI_Error*. В случае, когда значение любого параметра *out* является ненулевым, адаптер SA будет указывать на отсутствие ошибки. Следует отметить, что возвращаемое значение этой операции интерфейса TRI не приводит к выводу об успехе или ошибке вызова процедуры. Следует отметить, что факультативное значение тайм-аута, которое может быть задано в последовательности ATS TTCN-3 для операции вызова, *не* включено в сигнатуру операции `triCall`. Модуль TE должен адресовать это формирование вызова путем запуска таймера для операции вызова TTCN-3 в адаптере PA с отдельным вызовом операции интерфейса TRI, т. е. `triStartTimer`.
ПРИМЕЧАНИЕ. – Этого можно было бы достичь, например, путем порождения нового потока или процесса. Однако данная обработка этого вызова процедуры зависит от реализации модуля TE.	

5.5.4.4 Операция triReply (TE → SA)

Сигнатура	`TriStatusType triReply( in TriComponentIdType componentId,`
	`in TriPortIdType tsiPortId,`
	`in TriAddressType SUTaddress,`
	`in TriSignatureIdType signatureId,`
	`in TriParameterListType parameterList,`
	`in TriParameterType returnValue)`

Параметры In	`componentId` идентификатор отвечающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который адаптеру системы SUT посылается ответ `SUTaddress` адрес (факультативный) места назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3 `returnValue` (факультативное) закодированное возвращаемое значение вызова процедуры
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triReply`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию одноадресного ответа TTCN-3 на порте компонента, который был отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций ответа TTCN-3, если для тестового примера не был задан компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Все параметры процедуры *out* и *inout* и возвращаемое значение содержат закодированные значения. Список `parameterList` содержит параметры вызова процедуры. Эти параметры являются параметрами, описанными в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI. Если для сигнатуры процедуры в последовательности ATS TTCN-3 не был определен тип возврата, то для возвращаемого значения будет послано особое значение `null`.
Результат	При инициировании этой операции адаптер SA может выдать ответ для вызова процедуры, соответствующего идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triReply` будет возвращать значение ***TRI_OK*** при успешном выполнении этой операции; в противном случае будет возвращено значение ***TRI_Error***. В случае, когда значение любого параметра *in* или неопределенное возвращаемое значение отличны от нуля, адаптер SA будет указывать на отсутствие ошибки.

5.5.4.5 Операция triReplyBC (TE → SA)

Сигнатура	`TriStatusType triReplyBC( in TriComponentIdType componentId,`
	`in TriPortIdType tsiPortId,`
	`in TriSignatureIdType signatureId,`
	`in TriParameterListType parameterList,`
	`in TriParameterType returnValue)`

Параметры In	`componentId` идентификатор отвечающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который адаптеру системы SUT посылается ответ `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3 `returnValue` (факультативное) возвращаемое значение вызова процедуры
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triReply`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию широковещательного ответа TTCN-3 на порте компонента, который был отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций ответа TTCN-3, если для тестового примера не был задан компонент системы, т. е. для тестового примера создается только тестовый компонент MTC. Все параметры процедуры *out* и *inout* и возвращаемое значение содержат закодированные значения. Список `parameterList` содержит параметры вызова процедуры. Эти параметры являются параметрами, описанными в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI. Если для сигнатуры процедуры в последовательности ATS TTCN-3 не был определен тип возврата, то для возвращаемого значения будет послано особое значение `null`.
Результат	При инициировании этой операции адаптер SA может послать широковещательный ответ вызовам процедуры, соответствующим идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triReply` будет возвращать значение ***TRI_OK*** при успешном выполнении этой операции; в противном случае будет возвращено значение ***TRI_Error***. В случае, когда значение любого параметра *in* или неопределенное возвращаемое значение отличаются от нуля, адаптер SA будет указывать на отсутствие ошибки.

5.5.4.6 Операция triReplyMC (TE → SA)

Сигнатура	`TriStatusType triReplyMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriParameterListType parameterList, in TriParameterType returnValue)`
Параметры In	`componentId` идентификатор отвечающего тестового компонента `tsiPortId` идентификатор порта интерфейса тестовой системы, через который ответ посылается адаптеру системы SUT `SUTaddresses` адреса мест назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3 `returnValue` (факультативное) закодированное возвращаемое значение вызова процедуры
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triReply`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию многоадресного ответа TTCN-3 на порте компонента, который был отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций ответа TTCN-3, если для тестового примера не был задан компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Все параметры процедуры *out* и *inout* и возвращаемое значение содержат закодированные значения. Список `parameterList` содержит параметры вызова процедуры. Эти параметры являются параметрами, описанными в шаблоне сигнатуры TTCN-3. Их кодирование должно осуществляться в модуле TE до этого вызова операции интерфейса TRI. Если для сигнатуры процедуры в последовательности ATS TTCN-3 не был определен тип возврата, то для возвращаемого значения будет передано особое значение `null`.
Результат	При инициировании этой операции адаптер SA может передать многоадресный ответ вызовам процедуры, соответствующим идентификатору `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triReply` будет возвращать значение ***TRI_OK*** при успешном выполнении этой операции; в противном случае она будет возвращать значение ***TRI_Error***. В случае, если значение любого параметра *in* или неопределенное возвращаемое значение отличаются от нуля, адаптер SA будет указывать на отсутствие ошибки.

5.5.4.7 Операция triRaise (TE → SA)

Сигнатура	`TriStatusType triRaise( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriSignatureIdType signatureId, in TriExceptionType exc)`
Параметры In	`componentId` идентификатор тестового компонента, порождающего особое состояние `tsiPortId` идентификатор порта интерфейса тестовой системы, через который особое состояние передается адаптеру системы SUT `SUTaddress` адрес (факультативный) места назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры, с которым связано особое состояние `exc` закодированное особое состояние
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triRaise`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию одноадресного порождения особого состояния TTCN-3 на порте компонента, который был отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций порождения особого состояния TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создается только тестовый компонент MTC. Кодирование особого состояния должно быть выполнено в модуле TE перед этим вызовом операции интерфейса TRI.
Результат	При инициализации этой операции адаптер SA может породить особое состояние для вызова процедуры, соответствующего идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triRaise` возвращает значение *TRI_OK* при успешном выполнении операции; в противном случае она возвращает значение *TRI_Error*.

5.5.4.8 Операция triRaiseBC (TE → SA)

Сигнатура	`TriStatusType triRaiseBC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriSignatureIdType signatureId, in TriExceptionType exc)`
Параметры In	`componentId` идентификатор тестового компонента, порождающего особое состояние `tsiPortId` идентификатор порта интерфейса тестовой системы, через который особое состояние посылается адаптеру системы SUT `signatureId` идентификатор сигнатуры вызова процедуры, с которым связано особое состояние `exc` закодированное особое состояние
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triRaise`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию широковещательной передачи порождения особого состояния TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций порождения особого состояния TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Кодирование особого состояния должно осуществляться в модуле TE перед этим вызовом операции интерфейса TRI.
Результат	При инициировании этой операции адаптер SA может порождать и осуществлять широковещательную передачу особого состояния к вызовам процедуры, соответствующим идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triRaise` возвращает *TRI_OK* при успешном выполнении операции; в противном случае она возвращает значение *TRI_Error*.

5.5.4.9 Операция triRaiseMC (TE → SA)

Сигнатура	`TriStatusType triRaiseMC( in TriComponentIdType componentId, in TriPortIdType tsiPortId, in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId, in TriExceptionType exc)`
Параметры In	`componentId` идентификатор тестового компонента, порождающего особое состояние `tsiPortId` идентификатор порта интерфейса тестовой системы, через который особое состояние посылается адаптеру системы SUT `SUTaddresses` адреса мест назначения в системе SUT `signatureId` идентификатор сигнатуры вызова процедуры, с которым связано особое состояние `exc` закодированное особое состояние
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triRaise`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию многоадресной передачи порождения особого состояния TTCN-3 на порте компонента, который отображен на порт интерфейса TSI. Эта операция вызывается модулем TE для всех операций порождения особого состояния TTCN-3, если для тестового примера не был определен компонент системы, т. е. для тестового примера создан только тестовый компонент MTC. Кодирование особого состояния должно выполняться в модуле TE до этого вызова операции интерфейса TRI.
Результат	При инициировании этой операции адаптер SA может порождать и выполнять многоадресную передачу особого состояния вызовам процедуры, соответствующим идентификатору сигнатуры `signatureId` и идентификатору `tsiPortId` порта интерфейса TSI. Операция `triRaise` возвращает значение *TRI_OK* при успешном выполнении операции; в противном случае она возвращает значение *TRI_Error*.

5.5.4.10 Операция triEnqueueCall (SA → TE)

Сигнатура	`void triEnqueueCall( in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriComponentIdType componentId, in TriSignatureIdType signatureId, in TriParameterListType parameterList)`
Параметры In	`tsiPortId` идентификатор порта интерфейса тестовой системы, через который вызов процедуры ставится в очередь адаптером системы SUT `SUTaddress` (факультативный) адрес источника в системе SUT `componentId` идентификатор принимающего тестового компонента `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3. Описание данных, пересылаемых в качестве параметров к операции от вызывающего объекта к вызываемому объекту.
Параметры Out	не применяется
Возвращаемое значение	пустое значение
Ограничения	Эта операция может быть вызвана адаптером SA после того, как он получил от системы SUT вызов процедуры. Она может быть использована только тогда, когда идентификатор `tsiPortId` был ранее отображен на порт идентификатора `componentId`, либо на него была сделана ссылка в предыдущем операторе `triExecuteTestCase`. При инициировании операции `triEnqueueCall` все параметры процедуры *in* и *inout* содержат закодированные значения.
Результат	Модуль TE может поставить в очередь этот вызов процедуры с идентификатором сигнатуры `signatureId` на порте компонента `componentId`, на который отображен порт интерфейса TSI `tsiPortId`. Декорирование параметров процедуры должно быть выполнено в модуле TE. Модуль TE будет указывать на отсутствие ошибки в случае, когда значение любого параметра *out* отлично от нуля.

5.5.4.11 Операция triEnqueueReply (SA → TE)

Сигнатура	`void triEnqueueReply( in TriPortIdType tsiPortId, in TriAddressType SUTaddress, in TriComponentIdType componentId, in TriSignatureIdType signatureId, in TriParameterListType parameterList, in TriParameterType returnValue)`
Параметры In	`tsiPortId` идентификатор порта интерфейса тестовой системы, через который ответ ставится в очередь адаптером системы SUT `SUTaddress` (факультативный) адрес источника в системе SUT `componentId` идентификатор принимающего тестового компонента `signatureId` идентификатор сигнатуры вызова процедуры `parameterList` список закодированных параметров, являющихся частью указанной сигнатуры. Параметры в списке parameterList следуют в том порядке, в каком они располагаются в объявлении сигнатуры TTCN-3. `returnValue` (факультативное) закодированное возвращаемое значение вызова процедуры
Параметры Out	не применяется
Возвращаемое значение	пустое значение
Ограничения	Эта операция может быть вызвана адаптером SA после получения им ответа от системы SUT. Она может быть использована только тогда, когда идентификатор `tsiPortId` был ранее отображен на порт идентификатора `componentId`, либо на него была сделана ссылка в предшествующем операторе `triExecuteTestCase`. При инициировании операции `triEnqueueReply` все параметры процедуры *out* и *inout* и возвращаемое значение содержат закодированные значения. Если для сигнатуры процедуры в последовательности ATS TTCN-3 не был определен тип возврата, то для возвращаемого значения будет определено особое значение `null`.
Результат	Модуль TE может поставить в очередь этот ответ на вызов процедуры с идентификатором сигнатуры `signatureId` на порте компонента `componentId`, на который отображен порт `tsiPortId` интерфейса TSI. Кодирование параметров процедуры должно быть выполнено в модуле TE. Модуль TE будет указывать на отсутствие ошибки в случае, когда значение любого параметра *in* или неопределенное возвращаемое значение отличаются от нуля.

5.5.4.12 Операция triEnqueueException (SA → TE)

Сигнатура	`void triEnqueueException( in TriPortIdType tsiPortId,`
	`in TriAddressType SUTaddress,`
	`in TriComponentIdType componentId,`
	`in TriSignatureIdType signatureId,`
	`in TriExceptionType exc)`

Параметры In	`tsiPortId` идентификатор для порта интерфейса тестовой системы, через который особое состояние ставится в очередь адаптером системы SUT `SUTaddress` (факультативный) адрес источника в системе SUT `componentId` идентификатор принимающего тестового компонента `signatureId` идентификатор сигнатуры вызова процедуры, с которым связано особое состояние `exc` закодированное особое состояние
Параметры Out	не применяется
Возвращаемое значение	пустое значение
Ограничения	Эта операция может быть вызвана адаптером SA после получения им ответа от системы SUT. Она может быть использована только тогда, когда идентификатор `tsiPortId` был ранее отображен на порт идентификатора `componentId` или на него была сделана ссылка в предыдущем операторе `triExecuteTestCase`. При инициировании операции `triEnqueueException` особое состояние `exception` должно содержать закодированное значение.
Результат	Модуль TE может поставить в очередь это особое состояние для вызова процедуры с идентификатором процедуры `signatureId` на порте компонента `componentId`, на который отображается идентификатор `tsiPortId` порта интерфейса TSI. Декодирование особого состояния должно выполняться в модуле TE.

5.5.5 Смешанные операции

5.5.5.1 Операция triSUTactionInformal (TE → SA)

Сигнатура	`TriStatusType triSUTactionInformal(in string description)`
Параметры In	`description` неформальное описание действия, предпринимаемого в системе SUT
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triSUTactionInformal`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет операцию действия системы SUT TTCN-3, которая содержит только строку.
Результат	При инициировании этой операции адаптер SA будет инициировать описанные действия, предпринимаемые в системе SUT, например включать, инициировать или посылать сообщение системе SUT. Операция `triSUTactionInformal` возвращает значение *TRI_OK* при успешном выполнении операции; в противном случае возвращает значение *TRI_Error*. Следует заметить, что возвращаемое значение этой операции интерфейса TRI не приводит к выводу относительно успеха или ошибки действий, предпринимаемых в системе SUT.

5.5.5.2 Операция triSUTactionTemplate (TE → SA)

Устаревшая операция.

5.6 Операции интерфейса платформы

5.6.1 Операция triPAReset (TE → PA)

Сигнатура	`TriStatusType triPAReset()`
Параметры In	не применяется
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triPAReset`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция может быть вызвана модулем TE в любое время для сброса адаптера PA.
Результат	Адаптер PA будет сбрасывать все действия по синхронизации, которые он выполняет в текущее время, например останавливать все работающие таймеры, отбрасывать все значения тайм-аутов в состоянии ожидания для таймеров с истекшим временем действия. Операция `triResetSA` возвращает значение *TRI_OK* в случае, если операция выполнена успешно; в противном случае она возвращает значение *TRI_Error*.

5.6.2 Операции таймера

5.6.2.1 Операция triStartTimer (TE → PA)

Сигнатура	`TriStatusType triStartTimer( in TriTimerIdType timerId, in TriTimerDurationType timerDuration)`
Параметры In	`timerId` идентификатор экземпляра таймера `timerDuration` период времени таймера в секундах
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triStartTimer`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда требуется запустить таймер.
Результат	При инициировании этой операции адаптер PA будет запускать заданный таймер с указанным периодом времени. Таймер запускается с нулевого значения (0.0) и до максимума, заданного параметром timerDuration. Если таймер, на который указывает идентификатор `timerId`, уже работает, то он должен быть перезапущен. Когда период времени таймера истекает, адаптер PA вызывает операцию `triTimeout()` с идентификатором `timerId`. Операция `triStartTimer` возвращает значение ***TRI_OK***, если таймер был запущен успешно; в противном случае она возвращает значение ***TRI_Error***.

5.6.2.2 Операция triStopTimer (TE → PA)

Сигнатура	`TriStatusType triStopTimer( in TriTimerIdType timerId)`
Параметры In	`timerId` идентификатор экземпляра таймера
Параметры Out	не применяется
Возвращаемое значение	Состояние возврата операции `triStopTimer`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда требуется остановить таймер.
Результат	При инициировании этой операции адаптер PA будет использовать идентификатор `timerId` для остановки указанного экземпляра таймера. Останов неактивного таймера, т. е. таймера, который не запущен, или таймера с уже истекшим временем действия, не должен оказывать никакого влияния. Операция `triStopTimer` возвращает значение ***TRI_OK***, если операция была выполнена успешно; в противном случае она возвращает значение ***TRI_Error***. Следует заметить, что останов неактивного таймера является допустимой операцией. В этом случае значение ***TRI_OK*** должно быть возвращено.

5.6.2.3 Операция triReadTimer (TE → PA)

Сигнатура	`TriStatusType triReadTimer( in TriTimerIdType timerId, out TriTimerDurationType elapsedTime)`
Параметры In	`timerId` идентификатор экземпляра таймера
Параметры Out	`elapsedTime` значение истекшего времени в секундах с момента запуска таймера
Возвращаемое значение	Состояние возврата операции `triReadTimer`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция может вызываться модулем TE, когда операция считывания таймера TTCN-3 должна быть выполнена в указанном таймере (см. п. 5.3.1).
Результат	При инициировании этой операции адаптер PA будет использовать идентификатор `timerId` для доступа ко времени, которое истекло с момента запуска этого таймера. Возвращаемое значение `elapsedTime` будет выражено в секундах. Считывание неактивного таймера, т. е. таймера, который не был запущен или время действия которого уже истекло, будет возвращать нулевое значение истекшего периода времени. Операция `triReadTimer` возвращает значение ***TRI_OK***, если операция была выполнена успешно; в противном случае она возвращает значение ***TRI_Error***.

5.6.2.4 Операция triTimerRunning (TE → PA)

Сигнатура	`TriStatusType triTimerRunning( in TriTimerIdType timerId, out boolean running)`
Параметры In	`timerId` идентификатор экземпляра таймера
Параметры Out	`running` состояние таймера
Возвращаемое значение	Состояние возврата операции `triTimerRunning`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция может быть вызвана модулем TE, когда операция работы таймера TTCN-3 должна быть выполнена на указанном таймере (см. п. 5.3.1).
Результат	При инициировании этой операции адаптер PA будет использовать идентификатор `timerId` для доступа к состоянию таймера. Операция устанавливает параметр `running` на булево значение `true` тогда и только тогда, когда таймер в текущее время работает. Операция `triTimerRunning` возвращает значение ***TRI_OK***, если состояние таймера определено успешно; в противном случае она возвращает значение ***TRI_Error***.

5.6.2.5 Операция triTimeout (PA → TE)

Сигнатура	`void triTimeout(in TriTimerIdType timerId)`
Параметры In	`timerId` идентификатор экземпляра таймера
Параметры Out	Не применяется
Возвращаемое значение	пустое значение
Ограничения	Эта операция вызывается адаптером PA после того, как время действия таймера, ранее запущенного с использованием операции `triStartTimer`, истекло, т. е. оно достигло своего максимального значения для периода времени работы таймера.
Результат	Значение тайм-аута с идентификатором `timerId` может быть внесено в список значений тайм-аутов в модуле TE. Реализация этой операции в модуле TE должна быть выполнена таким образом, чтобы она адресовала разную семантику TTCN-3 для таймеров, определяемых в Рекомендации МСЭ-Т Z.143 [4] (см. также п. 5.3.1).

5.6.3 Смешанные операции

5.6.3.1 Операция triExternalFunction (TE → PA)

Сигнатура	`TriStatusType triExternalFunction( in TriFunctionIdType functionId, inout TriParameterListType parameterList, out TriParameterType returnValue)`
Параметры In	`functionId` идентификатор внешней функции
Параметры Out	`returnValue` (факультативное) закодированное возвращаемое значение
InOutParameters	`parameterList` список закодированных параметров для указанной функции. Параметры в списке `parameterList` следуют в том порядке, в каком они следуют в объявлении функции TTCN-3.
Возвращаемое значение	Состояние возврата операции `triExternalFunction`. Состояние возврата указывает на локальный успех (***TRI_OK***) или ошибку (***TRI_Error***) операции.
Ограничения	Эта операция вызывается модулем TE, когда он выполняет функцию, которая определяется в TTCN-3 как внешняя (т. е. все не внешние функции реализуются в модуле TE). При инициировании операции `triExternalFunction` модулем TE все параметры функции *in* и *inout* содержат закодированные значения. В случае ненулевого значения любого параметра *out* адаптер PA указывает на отсутствие ошибки.
Результат	Для каждой внешней функции, заданной в последовательности ATS TTCN-3, адаптер PA реализует поведение системы. При инициировании этой операции адаптер PA будет инициировать функцию, на которую указывает идентификатор `functionId`. Он будет осуществлять доступ к заданным параметрам функции *in u inout* в списке `parameterList`, оценивать внешнюю функцию, используя значения этих параметров, и вычислять значения для параметров *inout u out* в списке `parameterList`. Операция будет затем возвращать закодированные значения для всех параметров функции *inout* и *out* и закодированное возвращаемое значение внешней функции. Если для данной внешней функции в последовательности ATS TTCN-3 не был определен тип возврата, то для него должно быть использовано особое значение `null`. Операция `triExternalFunction` возвращает значение ***TRI_OK***, если адаптер PA завершает оценку внешней функции успешно; в противном случае она возвращает значение ***TRI_Error***. Следует заметить, что несмотря на то, что все прочие операции интерфейса TRI рассматриваются как неблокирующие, считается, что операция `triExternalFunction` является *блокирующей.* Это означает, что операция не должна возвращать значение, прежде чем указанная внешняя функция не будет полностью оценена. Внешние функции должны реализовываться с осторожностью, поскольку они могут вызывать блокировку выполнения тестового компонента и даже заблокировать реализацию всей тестовой системы.

6 Отображение языка Java

6.1 Введение

В данном разделе представлено отображение языка Java интерфейса TRI. Из-за большей эффективности вместо использования отображения языка IDL стандарта OMG на язык Java используется специализированное отображение языка.

В отображении языка Java для интерфейса времени выполнения TTCN-3 дается описание того, каким образом IDL-определения, приведенные в разделе 5, отображаются на язык Java. Отображение языка не зависит от используемой версии языка Java, поскольку используются только базовые конструкции этого языка.

6.2 Имена и контексты

6.2.1 Имена

К идентификаторам IDL применяются некоторые правила трансляции имен, хотя между идентификаторами, используемыми в определении IDL и в языке Java, конфликт отсутствует.

- Идентификаторы параметров Java будут начинаться со строчной буквы, а последующие части, образующие идентификатор параметра, будут начинаться с прописной буквы. Например, идентификатор **`SUTaddress`** параметра IDL отображается на `sutAddress` в языке Java.
- В интерфейсах или идентификаторах класса в языке Java опускается конечный `Type`, используемый в IDL-определении. Например, тип **`TriPortIdType`** в IDL отображается на `TriPortId` в языке Java.

Полученное в результате отображение отвечает стандартным правилам кодирования в языке Java.

6.2.2 Контексты

Модуль **`triInterface`** в IDL отображается на пакет `org.etsi.ttcn3.tri` в языке Java. Все IDL-описания типа в этом модуле отображаются на классы или описания интерфейсов на языке Java в рамках этого пакета.

6.3 Отображение типов

6.3.1 Отображение базовых типов

В таблице 3 дается обзор того, как используемые базовые IDL-типы отображаются на типы в языке Java.

Таблица 3/Z.144 – Отображения базовых типов

IDL-тип	Тип на языке Java
`boolean`	`org.etsi.ttcn.tri.TriBoolean`
`string`	`java.lang.String`

Другие базовые типы на языке IDL не используются в IDL-определении.

6.3.1.1 Булев тип

IDL-тип **`boolean`** отображается на интерфейс `org.etsi.ttcn.tri.TriBoolean`, так что объекты, реализующие этот интерфейс, могут действовать в качестве объектов-держателей.

Для `org.etsi.ttcn.tri.TriBoolean` определяется следующий интерфейс:

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

6.3.1.1.1 Методы

- `setBooleanValue(boolean value)`
  Устанавливает этот пакет `TriBoolean` по булеву значению `value`.

- `getBooleanValue()`
  Возвращает булево значение, представленное этим пакетом `TriBoolean`.

6.3.1.2 Строка

IDL-тип **`string`** отображается на класс `java.lang.String` без проверки диапазона или границ для символов в строке. Все возможные строки, заданные в TTCN-3, могут быть преобразованы в класс `java.lang.String`.

6.3.2 Отображение структурных типов

Типы, определяемые пользователем, даны в IDL-описании интерфейса TRI в качестве собственных типов. В отображении языка Java эти типы отображаются на Java-интерфейсы. Эти интерфейсы определяют методы и атрибуты, доступные для объектов, реализующих этот интерфейс.

6.3.2.1 Тип TriPortIdType

Тип **`TriPortIdType`** отображается на следующий интерфейс:

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

6.3.2.1.1 Методы

`- getPortName()`
Возвращает имя порта, как определено в спецификации TTCN-3.

`- getComponent()`
Возвращает идентификатор компонента, к которому принадлежит этот `TriPortId`, как определено в спецификации TTCN-3.

`- isArray()`
Возвращает `true`, если этот порт является частью массива портов, и `false` в противном случае.

`- getPortIndex()`
Возвращает индекс порта, если этот порт является частью массива портов, начиная с нуля. Если этот порт не является частью массива портов, тогда возвращается значение –1.

6.3.2.2 Тип TriPortIdListType

Тип **`TriPortIdListType`** отображается на следующий интерфейс:

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

6.3.2.2.1 Методы

`size()`
Возвращает число портов в этом списке.

`isEmpty()`
Возвращает `true`, если в этом списке нет портов.

`getPortIds()`
Возвращает `Enumeration` по всем портам в списке. При перечислении порты следуют в том же порядке, в каком они представлены в списке.

`get(int index)`
Возвращает `TriPortId` на определенное место.

6.3.2.3 Тип TriComponentIdType

Тип **`TriComponentIdType`** отображается на следующий интерфейс:

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

6.3.2.3.1 Методы

- `getComponentId()`
  Возвращает представление этого однозначно определяемого компонентного идентификатора.
- `getComponentName()`
  Возвращает имя компонента, как определено в спецификации TTCN-3. Если имя не предусмотрено, то возвращается пустая строка.
- `getComponentTypeName()`
  Возвращает имя типа компонента, как определено в спецификации TTCN-3.
- `getPortList()`
  Возвращает список портов компонента, как определено в спецификации TTCN-3.
- `equals(TriComponentId component)`
  Сравнивает `component` с этим идентификатором `TriComponentId`. Возвращает `true` тогда и только тогда, когда оба компонента обладают одним и тем же представлением этого однозначно определяемого идентификатора компонента, и возвращает `false` в противном случае.

6.3.2.4 Тип TriComponentIdListType

Тип **`TriComponentIdListType`** отображается на следующий интерфейс:

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

6.3.2.4.1 Методы

`size()`
  Возвращает число компонентов в этом списке.

`isEmpty()`
  Возвращает `true`, если этот список не содержит компонентов.

`getComponents()`
  Возвращает `Enumeration` по всем компонентам в списке. При перечислении компоненты следуют в том же порядке, в каком они представлены в списке.

`get(int index)`
  Возвращает `TriComponentId` на определенное место.

`clear()`
  Устраняет все компоненты из этого списка `TriComponentIdList`.

`add(TriComponentId comp)`
  Прибавляет `comp` к концу этого списка `TriComponentIdList`.

6.3.2.5 Тип TriMessageType

Тип **`TriMessageType`** отображается на следующий интерфейс:

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

6.3.2.5.1 Методы

- `getEncodedMessage()`
  Возвращает сообщение, закодированное согласно правилам кодирования, определенным в спецификации TTCN-3.
- `setEncodedMessage(byte[] message)`
  Устанавливает представление закодированного сообщения этого сообщения `TriMessage` на `message`.
- `equals(TriMessage message)`
  Сравнивает `message` с этим сообщением `TriMessage` для определения равенства. Возвращает `true` тогда и только тогда, когда оба сообщения имеют одно и то же закодированное представление; в противном случае воззвращает `false`.

6.3.2.6 Тип TriAddressType

Тип **`TriAddressType`** отображается на следующий интерфейс:

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

6.3.2.6.1 Методы

- `getEncodedAddress()`
  Возвращает закодированный адрес.
- `setEncodedAddress(byte[] address)`
  Устанавливает закодированный адрес этого `TriAddress` на `address`.
- `equals(TriAddress address)`
  Сравнивает `address` с этим `TriAddress` для определения равенства. Возвращает `true` тогда и только тогда, когда оба адреса имеют одно и то же закодированное представление; в противном случае возвращает `false`.

6.3.2.7 Тип TriAddressListType

Тип **`TriAddressListType`** отображается на следующий интерфейс:

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

6.3.2.7.1 Методы

`size()`
  Возвращает число компонентов в этом списке.

`isEmpty()`
  Возвращает `true`, если этот список не содержит компонентов.

`getAddresses()`
Возвращает `Enumeration` по всем компонентам в списке. При перечислении адреса следуют в том же порядке, в каком они представлены в списке.

`get(int index)`
Возвращает `TriAddress` на определенное место.

`clear()`
Удаляет все адреса из этого списка `TriAddressList`.

`add(TriAddress addr)`
Прибавляет `addr` к концу этого списка `TriAddressList`.

6.3.2.8 Тип TriSignatureIdType

Тип **`TriSignatureIdType`** отображается на следующий интерфейс:

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

6.3.2.8.1 Методы

- `getSignatureName()`
  Возвращает идентификатор сигнатуры, как определено в спецификации TTCN-3.

- `setSignatureName(String sigName)`
  Устанавливает идентификатор сигнатуры этого идентификатора `TriSignatureId` на `sigName`.

- `equals(TriSignatureId sig)`
  Сравнивает `sig` с этим идентификатором `TriSignatureId` для определения равенства. Возвращает `true` тогда и только тогда, когда обе сигнатуры имеют один и тот же идентификатор сигнатуры; в противном случае возвращает `false`.

6.3.2.9 Тип TriParameterType

Тип **`TriParameterType`** отображается на следующий интерфейс:

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

6.3.2.9.1 Методы

- `getParameterName()`
  Возвращает имя параметра, как определено в спецификации TTCN-3.

- `setParameterName(String name)`
  Устанавливает имя этого параметра `TriParameter` на `name`.

- `getParameterPassingMode()`
  Возвращает режим пересылки параметра для этого параметра.

- `setParameterPassingMode(in mode)`
  Устанавливает режим параметра этого параметра `TriParameter` на `mode`.

- `getEncodedParameter()`
  Возвращает представление закодированного параметра этого параметра `TriParameter` или объекта `null`, если параметр содержит особое значение `null` (см. также п. 5.5.4.1).

- `setEncodedParameter(byte[] parameter)`
  Устанавливает представление закодированного параметра этого параметра `TriParameter` на `parameter`. Если будет установлено особое значение `null` для указания на то, что этот параметр не содержит значения, то `null` языка Java будет пересылаться в качестве `parameter` (см. также п. 5.5.4.1).

6.3.2.10 Тип TriParameterPassingModeType

Тип **`TriParameterPassingModeType`** отображается на следующий интерфейс:

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

6.3.2.10.1 Константы

- `TRI_IN`
  Будет использоваться для указания на то, что `TriParameter` является параметром `in`.
- `TRI_INOUT`
  Будет использоваться для указания на то, что `TriParameter` является параметром `inout`.
- `TRI_OUT`
  Будет использоваться для указания на то, что `TriParameter` является параметром `out`.

6.3.2.11 Тип TriParameterListType

Тип **`TriParameterListType`** отображается на следующий интерфейс:

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

6.3.2.11.1 Методы

`size()`
  Возвращает число параметров в этом списке.

`isEmpty()`
  Возвращает `true`, если этот список не содержит параметров.

`getParameters()`
  Возвращает `Enumeration` по всем параметрам в списке. При перечислении параметры следуют в том же порядке, в каком они представлены в списке.

`get(int index)`
  Возвращает параметр `TriParameter` на определенное место.

`clear()`
  Удаляет все параметры из этого списка `TriParameterList`.

`add(TriParameter parameter)`
  Прибавляет `parameter` к концу этого списка `TriParameterList`.

6.3.2.12 Тип TriExceptionType

Тип **`TriExceptionType`** отображается на следующий интерфейс:

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

6.3.2.12.1 Методы

- `getEncodedException()`
  Возвращает особое состояние, закодированное согласно правилам кодирования, определенным в спецификации TTCN-3.
- `setEncodedMessage(byte[] exc)`
  Устанавливает представление закодированного особого состояния этого `TriException` на `exc`.
- `equals(TriException exc)`
  Сравнивает `exc` с этим `TriException` для определения равенства. Возвращает `true` тогда и только тогда, когда оба особых состояния имеют одно и то же закодированное представление; в противном случае возвращает `false`.

6.3.2.13 Тип TriTimerIdType

Тип **`TriTimerIdType`** отображается на следующий интерфейс:

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

6.3.2.13.1 Методы

- `getTimerName()`
  Возвращает имя этого идентификатора таймера, как определено в спецификации TTCN-3. В случае неявных таймеров результат зависит от реализации (см. п. 4.1.2).
- `equals(TriTimerId timer)`
  Сравнивает `timer` с этим идентификатором `TriTimerId` для определения равенства. Возвращает `true` тогда и только тогда, когда оба идентификатора таймера представляют один и тот же таймер; в противном случае возвращает `false`.

6.3.2.14 Тип TriTimerDurationType

Тип **`TriTimerDurationType`** отображается на следующий интерфейс:

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

6.3.2.14.1 Методы

- `getDuration()`
  Возвращает период времени таймера в виде `double`.
- `setDuration(double duration)`
  Устанавливает период времени этого `TriTimerDuration` на `duration`.
- `equals(TriTimerDuration duration)`
  Сравнивает `duration` с этим `TriTimerDuration` для определения равенства. Возвращает `true` тогда и только тогда, когда оба имеют один и тот же период времени; в противном случае возвращает `false`.

6.3.2.15 Тип TriFunctionIdType

Тип **`TriFunctionIdType`** отображается на следующий интерфейс:

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

6.3.2.15.1 Методы

- `toString()`
  Возвращает представление строки функции, как определено в спецификации TTCN-3.

- `getFunctionName()`
  Возвращает идентификатор функции, как определено в спецификации TTCN-3.

- `equals(TriFunctionId fun)`
  Сравнивает `fun` с этим идентификатором `TriFunctionId` для определения равенства. Возвращает `true` тогда и только тогда, когда обе функции имеют один и тот же идентификатор функции; в противном случае возвращает `false`.

6.3.2.16 Тип TriTestCaseIdType

Тип **`TriTestCaseIdType`** отображается на следующий интерфейс:

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

6.3.2.16.1 Методы

- `toString()`
  Возвращает представление строки тестового примера, как определено в спецификации TTCN-3.

- `getTestCaseName()`
  Возвращает идентификатор тестового примера, как определено в спецификации TTCN-3.

- `equals(TriTestCaseId tc)`
  Сравнивает `tc` с этим идентификатором `TriTestCaseId` для определения равенства. Возвращает `true` тогда и только тогда, когда оба тестовых примера имеют один и тот же идентификатор тестового примера; в противном случае возвращает `false`.

6.3.2.17 Тип TriActionTemplateType

Устаревший параметр.

6.3.2.18 Тип TriStatusType

Тип **`TriStatusType`** отображается на следующий интерфейс:

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

6.3.2.18.1 Методы

- `toString()`
  Возвращает представление строки состояния.

- `getStatus()`
  Возвращает состояние этого `TriStatus`.

- `setStatus(int status)`
  Устанавливает состояние этого `TriStatus`.

- `equals(TriStatus status)`
  Сравнивает `status` с этим состоянием `TriStatus` для определения равенства. Возвращает `true` тогда и только тогда, когда они имеют одно и то же состояние; в противном случае возвращает `false`.

6.4 Константы

Отображение констант в языке Java описано. Все константы определяются как `public final static` и доступны из каждого объекта любого пакета объектов. Константы, которые определены в данном разделе, не определяются в разделе IDL. Просто они являются следствием спецификации IDL-типов интерфейса TRI, помеченных как собственные.

Для определения режима пересылки параметров для параметров TTCN-3 можно использовать следующие константы (см. также п. 6.3.2.10).

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

Значения экземпляров этих констант должны отражать режим пересылки параметров, описанный в сигнатурах процедур TTCN-3.

Для особого значения параметра `null` закодированное значение параметра будет установлено на `null` в языке Java.

Для указания на локальный успех метода будут использоваться следующие константы (см. также п. 6.3.2.18):

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

6.5 Отображение интерфейсов

В IDL-описании интерфейса TRI определяются два интерфейса: интерфейс **`triCommunication`** и интерфейс **`triPlatform`**. Операции определяются для различных направлений в рамках этого интерфейса, т. е. некоторые операции могут вызываться только выполняемым модулем TTCN-3 (TE) в адаптере системы (SA), в то время как другие операции могут вызываться только адаптером SA в модуле TE. Это отражается путем разделения IDL-интерфейсов интерфейса TRI на два субинтерфейса, при этом каждый получает суффикс от вызываемого объекта.

Таблица 4/Z.144 – Субинтерфейсы

Вызывающий/Вызываемый	TE	SA	PA
TE	-	`TriCommunicationSA`	`triPlatformPA`
SA	`TriCommunicationTE`	-	-
PA	`TriPlatformTE`	-	-

Поведение всех методов, определяемых в этих интерфейсах, должно соответствовать описанию в разделе 5.

6.5.1 Режим пересылки параметров Out и InOut

В режиме пересылки параметров `out` или `inout` используются следующие IDL-типы:

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

В случае, когда они используются в режиме пересылки параметров `out` или `inout`, объекты соответствующего класса будут пересылаться с вызовом метода. Тогда вызываемый объект может иметь доступ к методам, чтобы установить возвращаемые значения.

6.5.2 Интерфейс triCommunication

Интерфейс **`triCommunication`** разделяется на два субинтерфейса: `triCommunicationSA`, определяющий вызовы от модуля TE к адаптеру SA, и `triCommunicationTE`, определяющий вызовы от SA к TE.

6.5.2.1 Интерфейс triCommunicationSA

Интерфейс `triCommunicationSA` отображается на следующий интерфейс:

```
// TriCommunication
// TE -> SA
package org.etsi.ttcn.tri;
public interface TriCommunicationSA {
    // Операция сброса
    // Ссылка: Описание TRI, п. 5.5.1
    TriStatus triSAReset();

    // Операции обработки соединения
    // Ссылка: Описание TRI, п. 5.5.2.1
    public TriStatus triExecuteTestCase(TriTestCaseId
        testCaseId,TriPortIdList tsiPorts);
    // Ссылка: Описание TRI, п. 5.5.2.2
    public TriStatus triMap(TriPortId compPortId, TriPortId tsiPortId);
    // Ссылка: Описание TRI, п. 5.5.2.3
    public TriStatus triUnmap(TriPortId compPortId, TriPortId tsiPortId);
```

```
    // Операции связи на базе сообщений
    // Ссылка: Описание TRI, п. 5.5.3.1
    public TriStatus triSend(TriComponentId componentId, TriPortId tsiPortId,
            TriAddress sutAddress, TriMessage sendMessage);
    // Ссылка: Описание TRI, п. 5.5.3.2
    public TriStatus triSendBC(TriComponentId componentId, TriPortId tsiPortId,
            TriMessage sendMessage);
    // Ссылка: Описание TRI, п. 5.5.3.3
    public TriStatus triSendMC(TriComponentId componentId, TriPortId tsiPortId,
            TriAddressList addresses, TriMessage sendMessage);

    // Операции связи на базе процедур
    // Ссылка: Описание TRI, п. 5.5.4.1
    public TriStatus triCall(TriComponentId componentId,
            TriPortId tsiPortId, TriAddress sutAddress,
            TriSignatureId signatureId, TriParameterList parameterList);
    // Ссылка: Описание TRI, п. 5.5.4.2
    public TriStatus triCallBC(TriComponentId componentId,
            TriPortId tsiPortId,
            TriSignatureId signatureId, TriParameterList parameterList);
    // Ссылка: Описание TRI, п. 5.5.4.3
    public TriStatus triCallMC(TriComponentId componentId,
            TriPortId tsiPortId, TriAddressList sutAddresses,
            TriSignatureId signatureId, TriParameterList parameterList);

    // Ссылка: Описание TRI, п. 5.5.4.4
    public TriStatus triReply(TriComponentId componentId,
            TriPortId tsiPortId, TriAddress sutAddress,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);
    // Ссылка: Описание TRI, п. 5.5.4.5
    public TriStatus triReplyBC(TriComponentId componentId,
            TriPortId tsiPortId,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);
    // Ссылка: Описание TRI, п. 5.5.4.6
    public TriStatus triReplyMC(TriComponentId componentId,
            TriPortId tsiPortId, TriAddressList sutAddresses,
            TriSignatureId signatureId, TriParameterList parameterList,
            TriParameter returnValue);

    // Ссылка: Описание TRI, п. 5.5.4.7
    public TriStatus triRaise(TriComponentId componentId, TriPortId tsiPortId,
            TriAddress sutAddress,
            TriSignatureId signatureId,
            TriException exc);
    // Ссылка: Описание TRI, п. 5.5.4.8
    public TriStatus triRaiseBC(TriComponentId componentId, TriPortId tsiPortId,
            TriSignatureId signatureId,
            TriException exc);
    // Ссылка: Описание TRI, п. 5.5.4.9
    public TriStatus triRaiseMC(TriComponentId componentId,TriPortId tsiPortId,
            TriAddresses sutAddresses,
            TriSignatureId signatureId,
            TriException exc);

    // Смешанные операции
    // Ссылка: Описание TRI, п. 5.5.5.1
    public TriStatus triSutActionInformal(Описание строки);

}
```

6.5.2.2 Интерфейс triCommunicationTE

Интерфейс `triCommunicationTE` отображается на следующий интерфейс:

```
// TriCommunication
// SA -> TE
package org.etsi.ttcn.tri;
public interface TriCommunicationTE {
    // Операции связи на базе сообщений
    // Ссылка: Описание TRI, п. 5.5.3.4
    public void triEnqueueMsg(TriPortId tsiPortId,
            TriAddress sutAddress, TriComponentId componentId,
            TriMessage receivedMessage);

    // Операции связи на базе процедур
    // Ссылка: Описание TRI, п. 5.5.4.10
    public void triEnqueueCall(TriPortId tsiPortId,
```

```
            TriAddress SUTaddress, TriComponentId componentId,
            TriSignatureId signatureId, TriParameterList parameterList );

    // Ссылка: Описание TRI, п. 5.5.4.11
    public void triEnqueueReply(TriPortId tsiPortId, TriAddress sutAddress,
            TriComponentId componentId, TriSignatureId signatureId,
            TriParameterList parameterList, TriParameter returnValue);

    // Ссылка: Описание TRI, п. 5.5.4.12
    public void triEnqueueException(TriPortId tsiPortId,
            TriAddress sutAddress, TriComponentId componentId,
        TriSignatureId signatureId, TriException exc);
}
```

6.5.3 Интерфейс triPlatform

Интерфейс **`triPlatform`** разделяется на два субинтерфейса: интерфейс `triPlatformPA`, определяющий вызовы от модуля TE к адаптеру PA, и интерфейс `triPlatformTE`, определяющий вызовы от PA к TE.

6.5.3.1 Интерфейс TriPlatformPA

Интерфейс `triPlatformPA` отображается на следующий интерфейс:

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

    // Операции обработки таймера
    // Ссылка: Описание TRI, п. 5.6.2.1
    public TriStatus triStartTimer(TriTimerId timerId,
        TriTimerDuration timerDuration);

    // Ссылка: Описание TRI, п. 5.6.2.2
    public TriStatus triStopTimer(TriTimerId timerId);

    // Ссылка: Описание TRI, п. 5.6.2.3
    public TriStatus triReadTimer(TriTimerId timerId,
        TriTimerDuration elapsedTime);

    // Ссылка: Описание TRI, п. 5.6.2.4
    public TriStatus triTimerRunning(TriTimerId timerId,
        TriBoolean running);

    // Смешанные операции

    // Ссылка: Описание TRI, п. 5.6.3.1
    public TriStatus triExternalFunction(TriFunctionId functionId,
     TriParameterList parameterList, TriParameter returnValue);
}
```

6.5.3.2 Интерфейс TriPlatformTE

Интерфейс `triPlatformTE` отображается на следующий Java-интерфейс:

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

6.6 Факультативные параметры

В разделе 5.4 определено, что резервируемое значение должно использоваться для указания на отсутствие факультативного параметра. В языке Java отображение значения `null` должно использоваться для указания на отсутствие факультативного значения. Например, если в операции `triSend` параметр адреса будет опущен, то инициированием операции будет `triSend(componentId, tsiPortId, null, sendMessage)`.

6.7 Инициализация интерфейса TRI

Все методы являются нестатическими, т. е. операции могут вызываться только в объектах. Поскольку в настоящей Рекомендации не определяются стратегии конкретной реализации модуля TE, адаптеров SA и PA, то механизм, с помощью которого для TE, SA или PA можно узнать дескрипторы в соответствующих объектах, выходит за рамки этой Рекомендации.

Поставщики инструментальных средств должны обеспечивать разработчикам адаптеров SA и PA методы регистрации TE, SA и PA для их соответствующего партнера связи.

6.8 Обработка ошибок

В рамках данного отображения языка Java не определяются никакие дополнительные виды обработки ошибок, кроме обработки ошибок, описанной в разделе 5.2. В частности, не определяется механизм обработки особых состояний.

7 Отображение языка ANSI-C

7.1 Введение

В данном разделе определяется отображение языка ANSI-C интерфейса TRI для абстрактных типов данных, описанных в разделе 5.3. Для базовых IDL-типов отображение соответствует рекомендациям стандарта OMG.

7.2 Имена и контексты

Идентификаторы параметра C будут начинаться со строчной буквы, а последующие части, образующие идентификатор параметра, будут начинаться с прописной буквы. Например, параметр `SUTaddress` в языке IDL отображается на `sutAddress` в языке C.

В идентификаторах абстрактных типов данных в языке C опускается конечный тип `Type`, используемый в IDL-определении. Например, IDL-тип `TriPortIdType` отображается на `TriPortId` в языке C.

В более ранних спецификациях языка C однозначность определения идентификатора была ограничена старшими значащими 8 символами. Тем не менее в недавних спецификациях ANSI-C это ограничение распространено на 31 старший значащий символ. Помимо этого вопроса, в данном отображении не наблюдается каких-либо противоречий, связанных с именами или контекстами.

7.2.1 Отображение абстрактных типов

Абстрактные типы данных интерфейса TRI	Представление в ANSI-C	Примечания и комментарии
`TriAddress`	`BinaryString`	

`TriAddressList`	`typedef struct TriAddressList`
	`{`
	` TriAddress** addrList;`
	` long int length;`
	`} TriAddressList;`
		ПРИМЕЧАНИЕ. – Для пометки конца списка `addrList[]` нет специальных значений. Поле `length` будет использоваться для надлежащего прохождения этого массива.

`TriComponentId`	`typedef struct TriComponentId`
	`{`
	` BinaryString compInst;`
	` String compName;`
	` QualifiedName compType;`
	`} TriComponentId;`
		ПРИМЕЧАНИЕ. – `compInst` используется для экземпляра компонента.

`TriComponentIdList`	`typedef struct`
	`TriComponentIdList`
	`{`
	` TriComponentId** compIdList;`
	` long int length;`
	`} TriComponentIdList;`
		ПРИМЕЧАНИЕ. – Для пометки конца списка `compIdList[]` нет специальных значений. Поле `length` будет использоваться для надлежащего прохождения этого массива.

`TriException`	`BinaryString`	
`TriFunctionId`	`QualifiedName`	
`TriMessage`	`BinaryString`	

Абстрактные типы данных интерфейса TRI	Представление в ANSI-C	Примечания и комментарии
TriParameterList	typedef struct TriParameterList	ПРИМЕЧАНИЕ. – Для пометки конца списка parList нет специальных значений. Поле length будет использоваться для надлежащего прохождения этого массива.
TriParameter	typedef struct TriParameter { BinaryString par; TriParameterPassingMode mode; } TriParameter;	
TriParameterPassingMode	typedef enum { TRI_IN = 0, TRI_INOUT = 1, TRI_OUT = 2 } TriParameterPassingMode;	ПРИМЕЧАНИЕ. – Значения экземпляров этого типа будут отражать режим пересылки параметров, описанных в соответствующих сигнатурах процедур TTCN-3.
TriPortIdList	typedef struct TriPortIdList { TriPortId** portIdList; long int length; } TriPortIdList;	ПРИМЕЧАНИЕ. – Для пометки конца списка portIdList[] нет специальных значений. Поле length будет использоваться для надлежащего прохождения этого массива.
TriPortId	typedef struct TriPortId { TriComponentId compInst; char* portName; long int portIndex; QualifiedName portType; void* aux; } TriPortId;	ПРИМЕЧАНИЕ 1. – compInst используется для экземпляра компонента. ПРИМЕЧАНИЕ 2. – Значение индекса portIndex должно составлять –1 для вырожденного (нерегулярного) объявления. ПРИМЕЧАНИЕ 3. – Поле aux служит для расширения функциональных возможностей интерфейса TRI в будущем.
TriSignatureId	QualifiedName	
TriStatus	long int #define TRI_ERROR -1 #define TRI_OK 0	ПРИМЕЧАНИЕ. – Все отрицательные значения резервируются для расширения функциональных возможностей интерфейса TRI в будущем.
TriTestCaseId	QualifiedName	
TriTimerDuration	Double	
TriTimerId	BinaryString	ПРИМЕЧАНИЕ. – Оператор ожидания обработки в семантике таймера и отображении мгновенного состояния может влиять на представление в будущем.

7.2.2 Определение типов в языке ANSI-C

Абстрактные типы данных в языке C	Определение типа	Примечания и комментарии
BinaryString	typedef struct BinaryString { unsigned char* data; long int bits; void* aux; } BinaryString;	ПРИМЕЧАНИЕ 1. – data представляет собой строку без завершающего нуля. ПРИМЕЧАНИЕ 2. – bits указывает на число битов, используемых в данных. Для обозначения опущенного значения используется значение битов, равное –1. ПРИМЕЧАНИЕ 3. – Поле aux служит для расширения функциональных возможностей интерфейса TRI в будущем.
QualifiedName	typedef struct QualifiedName { char* moduleName; char* objectName; void* aux; } QualifiedName;	ПРИМЕЧАНИЕ 1. – Поля moduleName и objectName являются буквально идентификаторами TTCN-3. ПРИМЕЧАНИЕ 2. – Поле aux служит для расширения функциональных возможностей интерфейса TRI в будущем.

7.2.3 Отображение IDL-типов

IDL-тип	Представление в ANSI-C	Примечания и комментарии
`Boolean`	`unsigned char`	Отображение от IDL стандарта OMG к C++
`String`	`char*`	Отображение от IDL стандарта OMG к C++

7.2.4 Отображение операций интерфейса TRI

Представление в IDL	Представление в ANSI-C
`TriStatusType triSAReset()`	`TriStatus triSAReset()`

`TriStatusType triExecuteTestCase`
`(in TriTestCaseIdType testCaseId,`
`in TriPortIdListType tsiPortList)`
	`TriStatus triExecuteTestCase`
	`(const TriTestCaseId* testCaseId,`
	`const TriPortIdList* tsiPortList)`

`TriStatusType triMap`
`(in TriPortIdType compPortId,`
`in TriPortIdType tsiPortId)`
	`TriStatus triMap`
	`(const TriPortId* compPortId,`
	`const TriPortId* tsiPortId)`

`TriStatusType triUnmap`
`(in TriPortIdType compPortId,`
`in TriPortIdType tsiPortId)`
	`TriStatus triUnmap`
	`(const TriPortId* compPortId,`
	`const TriPortId* tsiPortId)`

`TriStatusType triSend`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriAddressType SUTaddress,`
`in TriMessageType sendMessage)`
	`TriStatus triSend`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriAddress* sutAddress,`
	`const TriMessage* sendMessage)`

`TriStatusType triSendBC`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriMessageType sendMessage)`
	`TriStatus triSendBC`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriMessage* sendMessage)`

`TriStatusType triSendMC`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriAddressListType SUTaddresses,`
`in TriMessageType sendMessage)`
	`TriStatus triSendMC`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriAddressList* sutAddresses,`
	`const TriMessage* sendMessage)`

`void triEnqueueMsg`
`(in TriPortIdType tsiPortId,`
`in TriAddressType SUTaddress,`
`in TriComponentIdType componentId,`
`in TriMessageType receivedMessage)`
	`void triEnqueueMsg`
	`(const TriPortId* tsiPortId,`
	`const TriAddress* sutAddress,`
	`const TriComponentId* componentId,`
	`const TriMessage* receivedMessage)`

`TriStatusType triCall`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriAddressType SUTaddress,`
`in TriSignatureIdType signatureId,`
`in TriParameterListType parameterList)`
	`TriStatus triCall`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriAddress* sutAddress,`
	`const TriSignatureId* signatureId,`
	`const TriParameterList* parameterList)`

`TriStatusType triCallBC`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriSignatureIdType signatureId,`
`in TriParameterListType parameterList)`
	`TriStatus triCallBC`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriSignatureId* signatureId,`
	`const TriParameterList* parameterList)`

`TriStatusType triCallMC`
`(in TriComponentIdType componentId,`
`in TriPortIdType tsiPortId,`
`in TriAddressListType SUTaddresses,`
`in TriSignatureIdType signatureId,`
	`TriStatus triCallMC`
	`(const TriComponentId* componentId,`
	`const TriPortId* tsiPortId,`
	`const TriAddressList* sutAddresses,`
	`const TriSignatureId* signatureId,`

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

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

7.3 Управление памятью

Устаревшая функция.

7.4 Обработка ошибок

Для этого отображения обработка ошибок не была определена.

8 Сценарии использования

В данном разделе содержатся сценарии использования, которые должны помочь пользователям интерфейса TRI и поставщикам инструментальных средств, обеспечивающим этот интерфейс, понимать семантику операций, описанных в настоящей Рекомендации.

Исходя из схем последовательности сообщений (MSC) определены три сценария. Тот или иной сценарий состоит из кодового фрагмента TTCN-3, в котором используются функции связи TTCN-3 с системой SUT, а также функции обработки таймеров. На схеме MSC показаны взаимодействия между объектами TE, SA и PA вместе с системой SUT.

Следует особенно отметить, что фрагменты TTCN-3 не являются полными, поскольку основная цель фрагментов состоит в использовании динамического поведения. Во всех представленных сценариях используется общая последовательность преамбулы операций интерфейса TRI, представленная на рисунке 2.

Следует отметить, что в схемах MSC, представленных в данном разделе, используются пары сообщений для моделирования каждой операции интерфейса TRI. Сообщение triMap на схеме MSC, за которым следует сообщение triMapOK, означает, например, что операция triMap интерфейса TRI инициирована модулем TE и она успешно возвращает значение от адаптера SA. Вызовы операций интерфейса TRI представлены с использованием абстрактных типов и значений, и они предназначены только для иллюстративных целей. Конкретное представление этих параметров в отдельном целевом языке определяется в соответствующих отображениях языка.

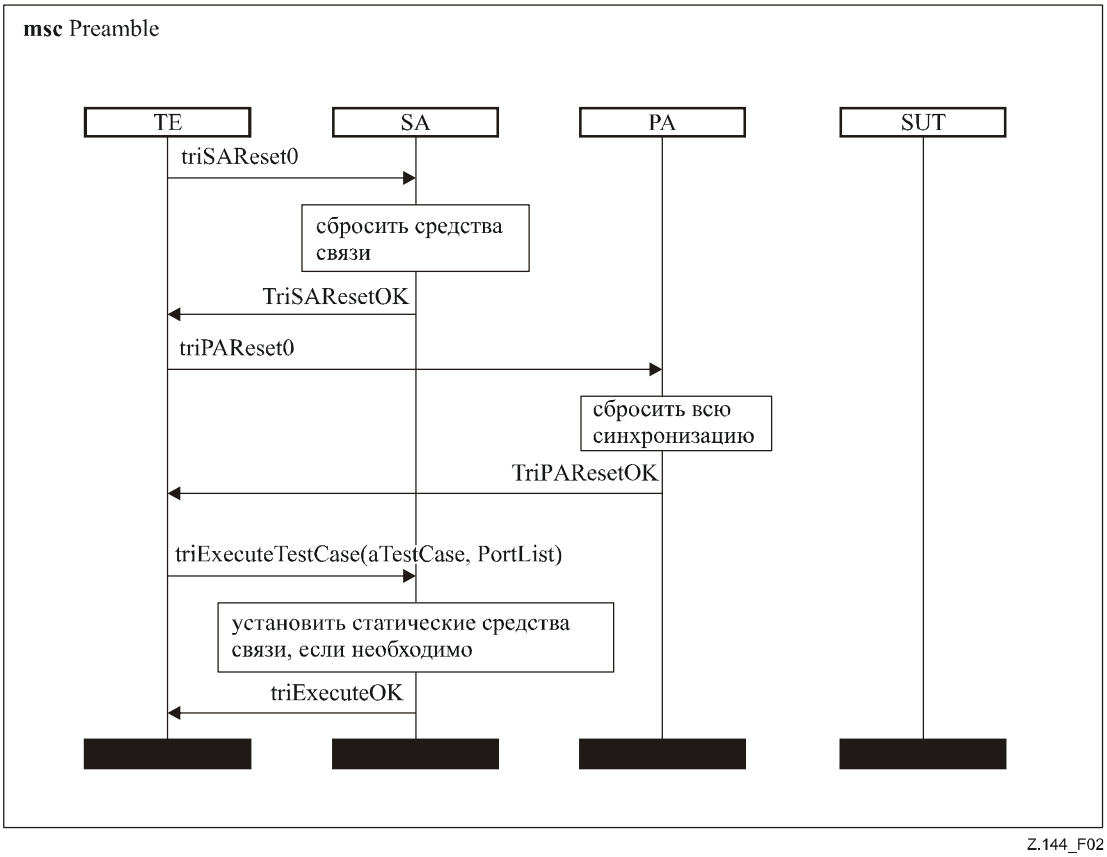

Рисунок 2/Z.144 – Общая преамбула схемы MSC

8.1 Первый сценарий

В первом сценарии показаны некоторые операции таймера TTCN-3, т. е. запуск и работа таймера, операции связи на базе сообщений, т. е. передача и прием, а также операции обработки соединения, т. е. отображение и неотображение.

8.1.1 Фрагмент TTCN-3

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

 type port PortTypeMsg message { inout integer }

 type component MyComponent {
    port PortTypeMsg MyPort;
    timer MyTimer
 }

 type component MyTSI {
    port PortTypeMsg PCO1;
 }

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

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

```
    MyPort.stop;
 }

 control {
    execute( scenario1() );
 }
}
```

8.1.2 Схема последовательности сообщений

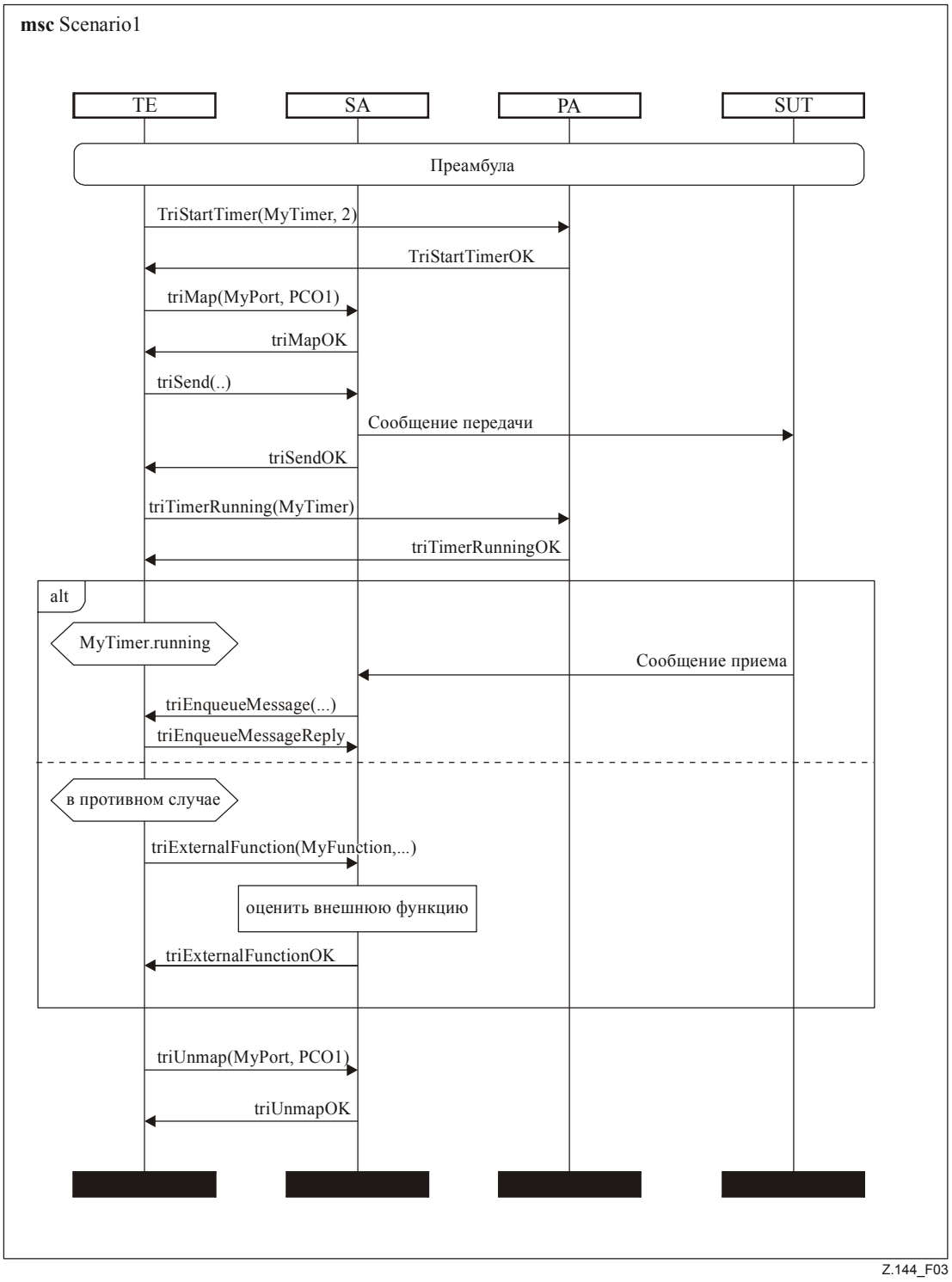

Рисунок 3/Z.144 – Сценарий 1 использования

8.2 Второй сценарий

Во втором примере представлен аналогичный сценарий, в котором также используются операции связи на базе процедур, инициируемых тестовым компонентом `MyComponent`. В этом примере предполагается, что `MyComponent` выполняется в качестве компонента МТС.

8.2.1 Фрагмент TTCN-3

```
module triScenario2
{

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

 type record MyExceptionType { FieldType1 par1, FieldType2 par2 }

 type port PortTypeProc procedure { out MyProc }

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

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

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

 MyVar := MyTimer.read;

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

 control {
    execute( scenario2() );
 }

}
```

8.2.2 Схема последовательности сообщений

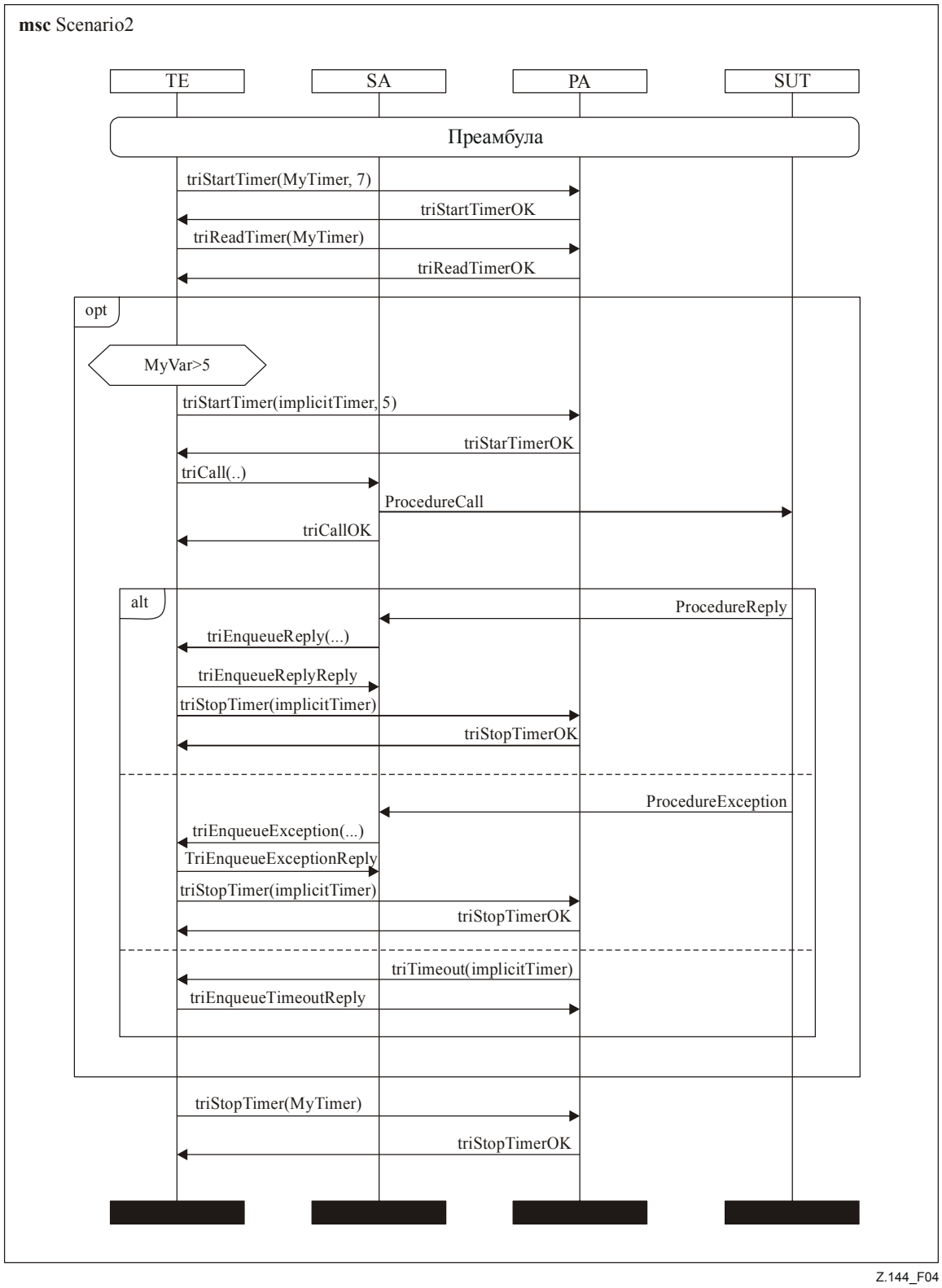

Рисунок 4/Z.144 – Сценарий 2 использования

8.3 Третий сценарий

В сценарии 3 использования представлены прием вызова процедуры, а также ответ и порождение особого состояния для этого принимаемого вызова. И снова предполагается, что `MyComponent` выполняется как компонент МТС. Предполагается, что `FieldType1`, `FieldType2`, `p1` и `p2` определены в другом месте.

8.3.1 Фрагмент TTCN-3

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

 type record MyExceptionType { FieldType1 par1, FieldType2 par2 }

 type port PortTypeProc procedure { in MyProc }

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

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

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

8.3.2 Схема последовательности сообщений

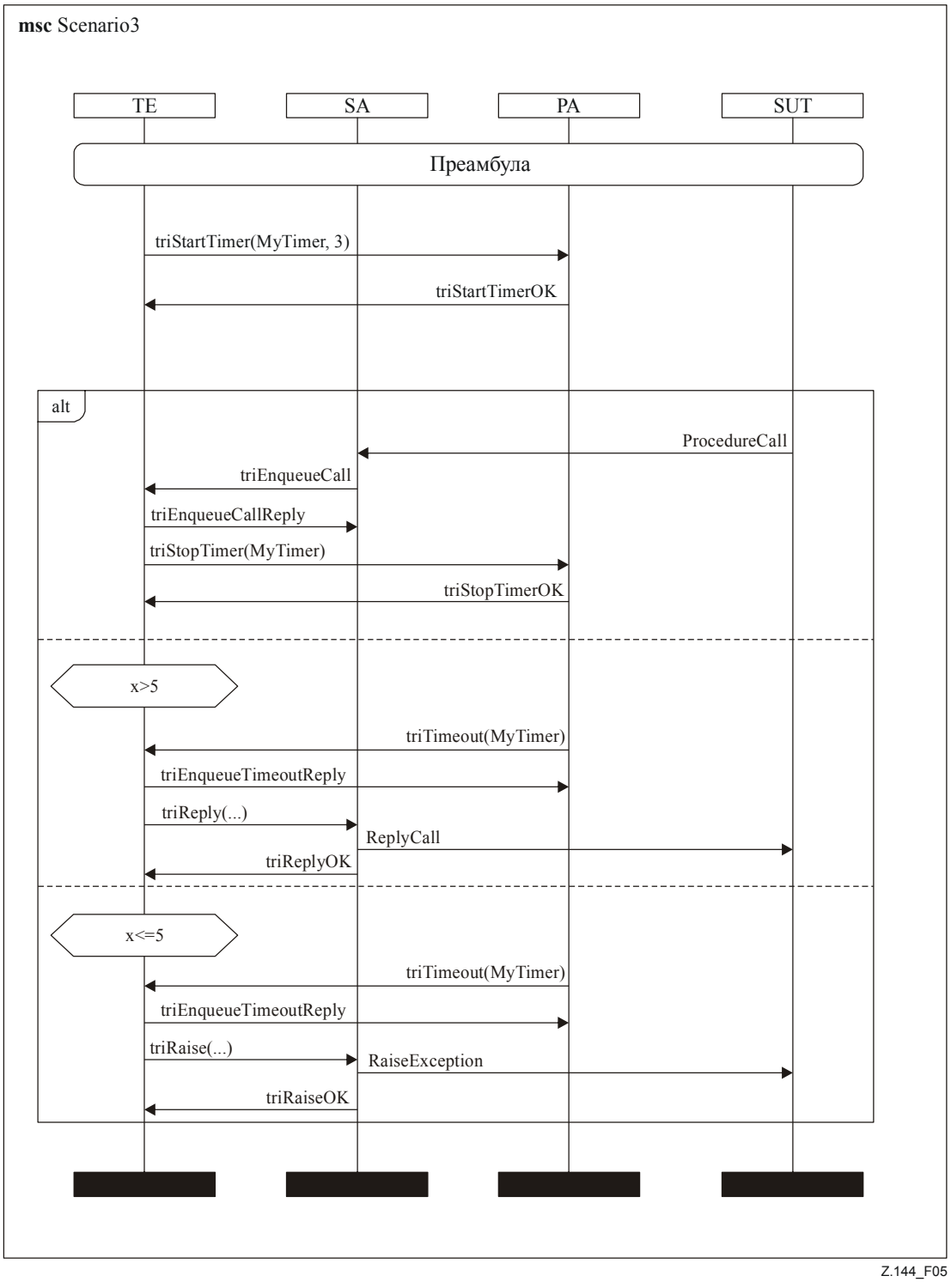

Рисунок 5/Z.144 – Сценарий 3 использования

Приложение A (нормативное)

Резюме IDL

В данном приложении резюмируется IDL-определение операций интерфейса TRI, как определено в разделе 5.

```
// ***************************************************************************
// Описание интерфейса для интерфейса времени выполнения TTCN-3

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

module triInterface
{

 //
 // ***************************************************************************
 // Типы
 // ***************************************************************************
 //

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

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

 // Синхронизация
 native TriTimerIdType;
 native TriTimerDurationType;

 // Смешанные
 native TriFunctionIdType;
 native TriTestCaseIdType;
 native TriStatusType;

 //
 // ***************************************************************************
 // Интерфейсы
 // ***************************************************************************
 //

 //
 // ***************************************************************************
 // Интерфейс связи (Ссылка: Описание TRI, п. 5.5)
 // ***************************************************************************
 //
 interface triCommunication
 {

    // Операция сброса

    // Ссылка: Описание TRI, п. 5.5.1
    TriStatusType triSAReset();


    // Операции обработки соединения

    // Ссылка: Описание TRI, п. 5.5.2.1
    TriStatusType triExecuteTestCase(in TriTestCaseIdType testCaseId,
 in TriPortIdListType tsiPortList);

    // Ссылка: Описание TRI, п. 5.5.2.2
    TriStatusType triMap(in TriPortIdType compPortId, in TriPortIdType tsiPortId);

    // Ссылка: Описание TRI, п. 5.5.2.3
    TriStatusType triUnmap(in TriPortIdType compPortId, in TriPortIdType tsiPortId);

    // Операции связи на базе сообщений
```

```
    // Ссылка: Описание TRI, п. 5.5.3.1
    TriStatusType triSend(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressType SUTaddress, in TriMessageType sendMessage);
    // Ссылка: Описание TRI, п. 5.5.3.2
    TriStatusType triSendBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriMessageType sendMessage);
    // Ссылка: Описание TRI, п. 5.5.3.3
    TriStatusType triSendMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressListType SUTaddresses, in TriMessageType sendMessage);

    // Ссылка: Описание TRI, п. 5.5.3.4
    void triEnqueueMsg(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
     in TriComponentIdType componentId, in TriMessageType receivedMessage);


    // Операции связи на базе процедур

    // Ссылка: Описание TRI, п. 5.5.4.1
    TriStatusType triCall(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressType SUTaddress, in TriSignatureIdType signatureId,
 in TriParameterListType parameterList);
    // Ссылка: Описание TRI, п. 5.5.4.2
    TriStatusType triCallBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriSignatureIdType signatureId,
 in TriParameterListType parameterList);
    // Ссылка: Описание TRI, п. 5.5.4.3
    TriStatusType triCallMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId,
 in TriParameterListType parameterList);

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

    // Ссылка: Описание TRI, п. 5.5.4.7
    TriStatusType triRaise(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressType SUTaddress, in TriSignatureIdType signatureId,
 in TriExceptionType exc);
    // Ссылка: Описание TRI, п. 5.5.4.8
    TriStatusType triRaiseBC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriSignatureIdType signatureId,
 in TriExceptionType exc);
    // Ссылка: Описание TRI, п. 5.5.4.9
    TriStatusType triRaiseMC(in TriComponentIdType componentId, in TriPortIdType tsiPortId,
 in TriAddressListType SUTaddresses, in TriSignatureIdType signatureId,
 in TriExceptionType exc);

    // Ссылка: Описание TRI, п. 5.5.4.10
    void triEnqueueCall(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
 in TriComponentIdType componentId, in TriSignatureIdType signatureId,
 in TriParameterListType parameterList );

    // Ссылка: Описание TRI, п. 5.5.4.11
    void triEnqueueReply(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
    in TriComponentIdType componentId, in TriSignatureIdType signatureId,
 in TriParameterListType parameterList, in TriParameterType returnValue );

    // Ссылка: Описание TRI, п. 5.5.4.12
    void triEnqueueException(in TriPortIdType tsiPortId, in TriAddressType SUTaddress,
    in TriComponentIdType componentId, in TriSignatureIdType signatureId,
 in TriExceptionType exc);

    // Смешанные операции

    // Ссылка: Описание TRI, п. 5.5.5.1
    TriStatusType triSUTactionInformal(в описании строки);

};
```

```
 //
 // *********************************************************************
 // Интерфейс платформы (Ссылка: Описание TRI, п. 5.6)
 // *********************************************************************
 //
 interface triPlatform
 {

    // Операция сброса

    // Ссылка: Описание TRI, п. 5.6.1
    TriStatusType triPAReset();


    // Операции обработки таймера

    // Ссылка: Описание TRI, п. 5.6.2.1
    TriStatusType triStartTimer(in TriTimerIdType timerId,
    in TriTimerDurationType timerDuration);

    // Ссылка: Описание TRI, п. 5.6.2.2
    TriStatusType triStopTimer(in TriTimerIdType timerId);

    // Ссылка: Описание TRI, п. 5.6.2.3
    TriStatusType triReadTimer(in TriTimerIdType timerId,
    out TriTimerDurationType elapsedTime);

    // Ссылка: Описание TRI, п. 5.6.2.4
    TriStatusType triTimerRunning(in TriTimerIdType timerId, out boolean running);

    // Ссылка: Описание TRI, п. 5.6.2.5
    void triTimeout(in TriTimerIdType timerId);


    // Смешанные операции

    // Ссылка: Описание TRI, п. 5.6.3.1
    TriStatusType triExternalFunction(in TriFunctionIdType functionId,
     inout TriParameterListType parameterList,
     out TriParameterType returnValue);

 };
};
```

Библиография

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

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

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия D	Общие принципы тарификации
Серия E	Общая эксплуатация сети, телефонная служба, функционирование служб и человеческие факторы
Серия F	Нетелефонные службы электросвязи
Серия G	Системы и среда передачи, цифровые системы и сети
Серия H	Аудиовизуальные и мультимедийные системы
Серия I	Цифровая сеть с интеграцией служб
Серия J	Кабельные сети и передача сигналов телевизионных и звуковых программ и других мультимедийных сигналов
Серия K	Защита от помех
Серия L	Конструкция, прокладка и защита кабелей и других элементов линейно-кабельных сооружений
Серия M	Управление электросвязью, включая СУЭ и техническое обслуживание сетей
Серия N	Техническое обслуживание: международные каналы передачи звуковых и телевизионных программ
Серия O	Требования к измерительной аппаратуре
Серия P	Качество телефонной передачи, телефонные установки, сети местных линий
Серия Q	Коммутация и сигнализация
Серия R	Телеграфная передача
Серия S	Оконечное оборудование для телеграфных служб
Серия T	Оконечное оборудование для телематических служб
Серия U	Телеграфная коммутация
Серия V	Передача данных по телефонной сети
Серия X	Сети передачи данных, взаимосвязь открытых систем и безопасность
Серия Y	Глобальная информационная инфраструктура, аспекты межсетевого протокола и сети последующих поколений
Серия Z	**Языки и общие аспекты программного обеспечения для систем электросвязи**

Отпечатано в Швейцарии
Женева, 2006 г.