МСЭ-Т

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

Q.834.4

(07/2003)

СЕРИЯ Q: КОММУТАЦИЯ И СИГНАЛИЗАЦИЯ

Интерфейс Q3

Спецификация интерфейса CORBA для широкополосных пассивных оптических сетей на основе требований интерфейса UML

Рекомендация МСЭ-Т Q.834.4

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ Q

КОММУТАЦИЯ И СИГНАЛИЗАЦИЯ

СИГНАЛИЗАЦИЯ В МЕЖДУНАРОДНОЙ СВЯЗИ С РУЧНОЙ КОММУТАЦИЕЙ	Q.1–Q.3
МЕЖДУНАРОДНАЯ АВТОМАТИЧЕСКАЯ И ПОЛУАВТОМАТИЧЕСКАЯ МЕЖДУНАРОДНАЯ СВЯЗЬ	Q.4–Q.59
ФУНКЦИИ И ИНФОРМАЦИОННЫЕ ПОТОКИ ДЛЯ СЛУЖБ В ЦСИС	Q.60–Q.99
СЛУЧАИ, ПРИМЕНИМЫЕ К СТАНДАРТИЗИРОВАННЫМ СИСТЕМАМ МСЭ-Т	Q.100–Q.119
ТРЕБОВАНИЯ К СИСТЕМАМ СИГНАЛИЗАЦИИ № 4, 5, 6, R1 И R2	Q.120–Q.499
ЦИФРОВЫЕ КОММУТАЦИОННЫЕ СТАНЦИИ	Q.500–Q.599
ВЗАИМОДЕЙСТВИЕ СИСТЕМ СИГНАЛИЗАЦИИ	Q.600–Q.699
ТРЕБОВАНИЯ К СИСТЕМЕ СИГНАЛИЗАЦИИ № 7	Q.700–Q.799
ИНТЕРФЕЙС Q3	**Q.800–Q.849**
ЦИФРОВАЯ АБОНЕНТСКАЯ СИСТЕМА СИГНАЛИЗАЦИИ № 1	Q.850–Q.999
СЕТЬ СУХОПУТНОЙ ПОДВИЖНОЙ СВЯЗИ ОБЩЕГО ПОЛЬЗОВАНИЯ	Q.1000–Q.1099
ВЗАИМОДЕЙСТВИЕ СО СПУТНИКОВЫМИ СИСТЕМАМИ ПОДВИЖНОЙ СВЯЗИ	Q.1100–Q.1199
ИНТЕЛЛЕКТУАЛЬНАЯ СЕТЬ	Q.1200–Q.1699
ТРЕБОВАНИЯ К СИГНАЛИЗАЦИИ И ПРОТОКОЛЫ IMT-2000	Q.1700–Q.1799
ТРЕБОВАНИЯ К СИГНАЛИЗАЦИИ, ОТНОСЯЩЕЙСЯ К УПРАВЛЕНИЮ ВЫЗОВОМ НЕЗАВИСИМО ОТ КАНАЛА-НОСИТЕЛЯ (BICC)	Q.1900–Q.1999
ШИРОКОПОЛОСНАЯ ЦСИС	Q.2000–Q.2999

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

Рекомендация МСЭ-Т Q.834.4

Спецификация интерфейса CORBA для широкополосных пассивных оптических сетей на основе требований интерфейса UML

Резюме

В данной Рекомендации приводится определение CORBA IDL для интерфейса управления между системой управления поставщика и системой управления оператора. Данная работа определяет часть аспектов управления сетевыми ресурсами, изложенных в Рекомендациях МСЭ-Т серии G.983.x для оборудования широкополосной пассивной оптической сети (BPON).

Вообще говоря, система управления поставщика – это система управления элементами (EMS), а система управления оператора (OMS) – это система управления сетью (NMS). Однако система управления поставщика нуждается в представлении системе управления оператора "сетевого взгляда" на управление соединениями. Поэтому было сочтено необходимым в целях большей ясности использовать терминологию, принятую в рассматриваемых системах.

Кроме того, следует отметить, что Рекомендация МСЭ-Т Q.834.1 содержит ряд функциональных требований и листинг управляемых объектов, формирующий основу информации управления, необходимой для "взгляда со стороны элемента сети" на оборудование BPON. Рекомендация МСЭ-Т Q.834.2 завершает определение информации управления оборудованием BPON, предлагая определения управляемых объектов с точки зрения "сетевого взгляда". Рекомендация МСЭ-Т Q.834.3 излагает поведение интерфейса управления посредством использования диаграмм UML и описаний случаев использования. Данная Рекомендация и Рекомендация МСЭ-Т Q.834.3 ссылаются на информацию управления, смоделированную в Рекомендациях МСЭ-Т Q.834.1 и Q.834.2.

Источник

Рекомендация МСЭ-Т Q.834.4 утверждена 7 июля 2003 года 4-й Исследовательской комиссией МСЭ-Т (2001–2004 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8. В данное издание включены изменения, содержащиеся в Поправке 1, утвержденной 13 января 2004 года.

Ключевые слова

APON, BPON, CORBA, IDL, PON, UML.

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.

Всемирная ассамблея по стандартизации электросвязи (ВАСЭ), которая проводится каждые четыре года, определяет 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.

Соответствие положениям данной Рекомендации является добровольным делом. Однако в Рекомендации могут содержаться определенные обязательные положения (для обеспечения, например, возможности взаимодействия или применимости), и тогда соответствие данной Рекомендации достигается в том случае, если выполняются все эти обязательные положения. Для выражения требований используются слова "shall" ("должен", "обязан") или некоторые другие обязывающие термины, такие как "must" ("должен"), а также их отрицательные эквиваленты. Использование таких слов не предполагает, что соответствие данной Рекомендации требуется от каждой стороны.

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

МСЭ обращает внимание на то, 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 не получил извещения об интеллектуальной собственности, защищенной патентами, которые могут потребоваться для реализации 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.

СОДЕРЖАНИЕ

Рекомендация МСЭ-Т Q.834.4

Спецификация интерфейса CORBA для широкополосных пассивных оптических сетей на основе требований интерфейса UML

1 Область применения

Данная Рекомендация относится к разработке интерактивного автоматизированного интерфейса между системой управления поставщика, управляющего ресурсами сети BPON, и системой управления оператора (OMS). Разработка основывается на диаграммах UML и описаниях пользовательского набора из Рекомендации МСЭ-Т Q.834.3. Общий подход, одобренный в Рекомендации МСЭ-Т Q.834.3 и реализованный в настоящем документе, – это подход, с помощью которого система управления поставщика обеспечивает системе управления оператора управляющие службы. Эти службы предоставляют OMS следующие возможности с высоким уровнем абстракции:

- инициализация установленных сетевых ресурсов;
- инициализация установленных сетевых ресурсов, включая резервирование пропускной способности;
- инициализация служб;
- управление архивом;
- управление программным обеспечением элементов сети;
- создание резервных копий и восстановление конфигурационных данных элементов сети;
- управление работой;
- публикация событий элементов сети;
- управление профилями;
- тестирование;
- планирование деятельности;
- управление передачей больших массивов данных;
- синхронизация элементов сети и системы управления элементами;
- управление доступом.

Данная Рекомендация описывает CORBA-интерфейсы IDL, поддерживающие службы, перечисленные выше.

2 Ссылки

2.1 Нормативные ссылки

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] ITU-T Recommendation M.3010 (2000), *Principles for a telecommunications management network*.

[2] ITU-T Recommendation M.3200 (1997), *TMN management services and telecommunications managed areas: Overview*.

[3] ITU-T Recommendation M.3400 (2000), *TMN Management Functions*.

[4] OMG Document formal/99-06-01, *Unified Modelling Language, Section 1*.

[5] ITU-T Recommendation G.983.1 (1998), *Broadband optical access systems based on Passive Optical Networks (PON)*, plus Amendment 1 (2001).

[6] ITU-T Recommendation Q.834.1 (2001), *ATM-PON requirements and managed entities for the network element view*.

[7] ITU-T Recommendation Q.834.2 (2001), *ATM-PON requirements and managed entities for the network view*.

[8] ITU-T Recommendation Q.834.3 (2001), *A UML description for management interface requirements for Broadband Passive Optical Networks*.

[9] ITU-T Recommendation M.3020 (2000), *TMN Interface Specification Methodology*.

[10] ITU-T Recommendation G.983.2 (2002), *ONT management and control interface specification for B-PON*.

[11] ITU-T Recommendation G.983.3 (2001), *A broadband optical access system with increased service capability by wavelength allocation*.

[12] ITU-T Recommendation G.983.4 (2001), *A broadband optical access system with increased service capability using dynamic bandwidth assignment (DBA)*.

[13] ITU-T Recommendation G.983.5 (2002), *A broadband optical access system with enhanced survivability*.

[14] ITU-T Recommendation G.983.6 (2002), *ONT management and control interface specifications for B-PON system with protection features*.

[15] ITU-T Recommendation G.983.7 (2001), *ONT Management and Control interface specification for Dynamic Bandwidth Assignment (DBA) B-PON system*.

[16] ITU-T Recommendation X.780 (2001), *TMN guidelines for defining CORBA managed objects*.

[17] ITU-T Recommendation Q.816 (2001), *CORBA-based TMN services*.

2.2 Другие ссылки

[18] OMG Document formal/02-09-02, *CORBA Services – Naming Service specification*.

[19] OMG Document formal/02-08-04, *CORBA Services – Notification Service specification*.

3 Термины и определения

3.1 Термины, взятые из Рекомендации МСЭ-Т M.3010

В данной Рекомендации используется следующий термин из Рекомендации МСЭ-Т M.3010:

– пользователь.

3.2 Термины, взятые из UML

В данной Рекомендации используются следующие термины из UML [4]:

– действующий субъект;

– класс;

– диаграмма класса;

– случай использования.

3.3 Термины, взятые из Службы присвоения имен OMG

В данной Рекомендации используются следующие термины из Службы присвоения имен OMG [18]:

– граф присваивания имен;

– имя.

3.4 Термины, взятые из Рекомендации МСЭ-Т Q.834.1

В данной Рекомендации используется следующий термин из Рекомендации МСЭ-Т Q.834.1:

– управляемый объект.

3.5 Термины, взятые из Рекомендации МСЭ-Т Q.834.3

В данной Рекомендации используются следующие термины из Рекомендации Q.834.3:

– активировать;

– присваивать;

– автоматическое раскрытие;

– ресурсы BPON;

– конструкция;

– фильтрация;

– инсталлировать;

– ранжировать;

– регистрировать;

– резервировать;

– экземпляр обслуживания;

– метка пользователя.

3.6 Новые термины

В данной Рекомендации определяются следующие термины:

3.6.1 объект обслуживания: Набор объектов, обеспечивающих доступ к службам управления, реализованным в системе управления поставщика. Интерфейсы к объектам обслуживания устанавливают спецификацию для части IF1, определенную в данной Рекомендации.

3.6.2 объект домена: Набор объектов, определяющих информацию для управления ресурсами сети BPON. Объекты домена первоначально задаются списками управляемых объектов Рекомендаций МСЭ-Т Q.834.1 и Q.834.2.

3.6.3 внутренний объект: Набор объектов, используемых для поддержки внутренней логики системы управления поставщика. С точки зрения OMS они полностью скрыты.

4 Сокращения

В данной Рекомендации используются следующие сокращения:

AAL	Уровень адаптации ATM
APON	ATM-PON
ATM	Асинхронный режим передачи
BICI	Широкополосный интерфейс соединительных линий между телефонными компаниями
BISSI	Широкополосный интерфейс между коммутационными системами
BPON	Широкополосная пассивная оптическая сеть

CAC	Управление доступом к вызову
CCITT	Международный консультативный комитет по телефонии и телеграфии (МККТТ)
CES	Услуга эмуляции цепи
CNM	Управление сетью пользователя
CORBA	Обобщенная архитектура с брокером (посредником) запросов к объектам
CTP	Точка окончания соединения
CTT	Метка проблемы пользователя
DCN	Сеть передачи данных
DSx	Цифровой сигнал x
EM	Управление элементом
EML	Уровень управления элементом
EMS	Система управления элементом
EOC	Встроенный эксплуатационный канал
Ex	Европейский цифровой сигнал x
FSAN	Сеть полного сервисного доступа
GUI	Графический интерфейс пользователя
IDL	Язык определения интерфейса
ITU	Международный союз электросвязи (МСЭ)
ME	Управляемый объект
MIB	База информации управления
NE	Сетевой элемент
NMS	Система сетевого управления
NT	Сетевое окончание
ODN	Сеть оптического распределения
OLT	Окончание оптической линии
OMG	Группа управления объектом
OMS	Система управления оператора
ONT	Окончание оптической сети
ONU	Оптическое сетевое устройство
OS	Операционная система
PON	Пассивная оптическая сеть
PVC	Постоянное виртуальное соединение
QoS	Качество обслуживания
TCA	Предупреждение о достижении пороговой величины
TMN	Сеть управления электросвязью
TP	Точка окончания
TTP	Оконечная точка трассы
UML	Унифицированный язык моделирования
UNI	Сетевой интерфейс пользователя
VC	Виртуальный канал
VCC	Соединение виртуального канала

VCI Идентификатор виртуального канала

VP Виртуальный путь

VPC Соединение по виртуальному пути

VPI Идентификатор виртуального пути

5 Соглашения

В данной Рекомендации есть два пункта, где действуют специфические соглашения. Первое – это описание модулей IDL в разделе 8, а второе – файлы IDL в Приложении C. Также в данной Рекомендации используются определенные соглашения, включающие в себя пустые значения.

5.1 Соглашения в описаниях модулей

Описания модулей имеют следующую структуру:

– Имя модуля – обзор служб, включая все соответствующие модели высокого уровня, описания элементов ключевых бизнес-данных и необходимые сценарии.

  • Имя интерфейса – описание конкретных услуг, предоставляемых данным интерфейсом.

    – Имя операции$_1$ – детальное описание поведения операции, включая сигнатуру, описания каждого входного параметра и возвращаемого значения.

    – …

    – Имя операции$_n$ – детальное описание поведения операции, включая сигнатуру, описания каждого входного параметра и возвращаемого значения.

    – Исключительные ситуации – зависящие от контекста условия, вызывающие возникновение указанных ситуаций.

5.2 Соглашения файлов IDL

Каждый файл IDL имеет следующий формат:

- **#ifndef __<MODULENAME>_DEFINED** (имя модуля то же самое, что и имя файла);
- **#define __<MODULENAME>_DEFINED**
- **#include "<idlfilename>"**[1]
- **#pragma prefix "itu.Int"**
- **module q834_4 {**
- **module <modulename> {**
- все указанные определения типов данных и спецификации интерфейса необходимы и задаются в ходе дизайна интерфейса, инициируемого в результате изучения Рекомендации МСЭ-Т Q.834.3;
- **}; //module <modulename>**
- **}; //module q834_4**

В каждом определении модуля в начале задаются определения типов данных, которые отделяются от определений интерфейса. Все типы данных, определяемые в настоящей Рекомендации, начинаются с символа заглавной буквы "альфа". Если имена типов данных состоят из нескольких существительных, местоимений или прилагательных, то заглавная буква "альфа" располагается в начале каждого компонента. Метки для параметров, используемые в сигнатуре любой операции интерфейса, начинаются со строчного символа "альфа". Каждое название операции также начинается со строчного символа "альфа".

---

[1] Файл Q834Common.idl включен в данную Рекомендацию вместе со всеми другими файлами IDL.

5.3 Пустые значение

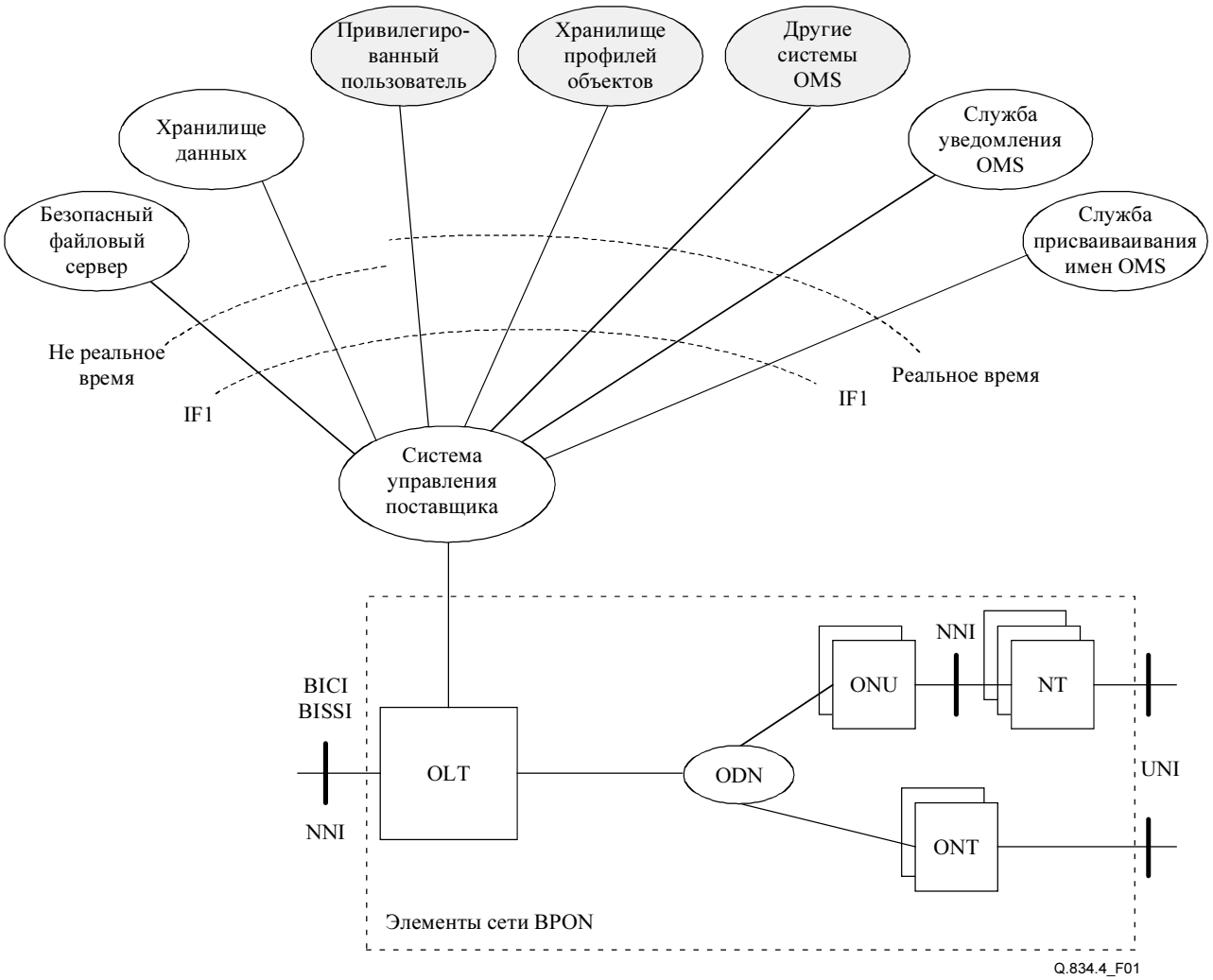
Когда в данной Рекомендации встречается упоминание "пустой строки", то оно означает пустую строку, а не отсутствующий объект, поддерживаемый такими языками, как JAVA. Аналогично, когда упоминается "пустая последовательность", то передается последовательность с нулевым количеством элементов.

6 Обзор архитектуры интерфейса

На рисунке 1 иллюстрируются архитектура и система управления сети BPON. Данная технология обеспечивает интегрированный циклический механизм предоставления доступа к каждой службе электросвязи, поддерживаемой операторами. Список таких служб включает телефонию и передачу голоса, эмуляцию цепи, Ethernet, ATM, xDSL, и видео. Эта стандартизация исходит из рассмотрения в сетях ODN транспортных решений уровня 1 и уровня 2, хотя первоначальные реализации, существующие в сетях операторов, в том числе и типа APON и ATM , определены в Рекомендации МСЭ-Т G.983.1.

На рисунке также показан интерфейс IF1 (Q) между системой управления поставщика и системой управления оператора. Интерфейс IF1 включает в себя все аспекты управления сетью, предоставлением услуг, работой сети, трафиком, а также вопросы технического обслуживания, тестирования и администрирования безопасности пользователей. Спецификация IF1 могла бы иметь выдающееся значение, если бы данный подход не занимал уровень абстракции выше, чем детали технологии транспортной доставки, а также если бы не сложность, свойственная управлению таким большим количеством типов служб. Подход, включающий в себя определение и описание требований интерфейса, использующего унифицированный язык моделирования, следует методологии Рекомендации МСЭ-Т M.3020 и документирован в Рекомендации МСЭ-Т Q.834.3.

Системы операторов, обозначенные на этой диаграмме, непосредственно соответствуют системным субъектам, определенным в Рекомендации МСЭ-Т Q.834.3. Данная Рекомендация определяет интерактивные транзакции реального времени, включающие в себя систему управления поставщика, и не дает ясного описания структуры файлов, передаваемых между системой управления поставщика и либо хранилищем данных, либо безопасным файловым сервером. Поскольку интерфейсы к службе присваивания имен и к службе уведомлений OMG уже определены, основное содержание данной Рекомендации сводится к описанию передачи сообщений между системой управления поставщика и скрытыми субъектами системы.



BICI	Широкополосный интерфейс между несущими	ODN	Сеть оптического распределения
BISSI	Широкополосный интерфейс между системами коммутации	OLT	Окончание оптической линии
BPON	Широкополосная пассивная оптическая сеть	ONT	Терминал оптической сети
NNI	Интерфейс между сетями	ONU	Оптическое сетевое устройство
NT	Сетевое окончание	UNI	Сетевой интерфейс пользователя

Рисунок 1/Q.834.4 – Обзор архитектуры интерфейса

7 Имена и ограничения в присваивании имен

В данном разделе определяются основные принципы присваивания имен и идентификаторов управляемых объектов в соответствии с IF1.

Обзор всех диаграмм классов, приведенных в Рекомендации МСЭ-Т Q.834.3, показывает, что система управления поставщика управляет тремя различными типами "объектов". Они могут быть классифицированы следующим образом:

- Служебные объекты: обеспечивают доступ к службам управления, реализованным в системе управления поставщика. Интерфейсы этих служб являются главным предметом данной Рекомендации. Примеры: NERegistrar, ServiceProvisioner, TestActionPerformer, и DownloadMgr. Они выстроены в данной Рекомендации в соответствии с Рекомендацией МСЭ-Т X.780 как производные интерфейса объектов управления.
- Объекты домена: определяют информацию, необходимую для управления ресурсами сети BPON. Объекты домена – это первичные ресурсы (управляемые объекты), определенные в Рекомендациях МСЭ-Т Q.834.1 и Q.834.2. Эти объекты, а также другие объекты, определенные в Рекомендации МСЭ-Т M.3120 и прочих Рекомендациях, относящихся к разделу CORBA, могут быть сделаны доступными для использования в IF1. Механизм гармонизации использования объектов служб и объектов домена более подробно объясняется далее в этом пункте.

- Внутренние объекты: используются для поддержки внутренней логики системы управления поставщика. Они полностью скрыты с точки зрения OMS.

7.1 Служебные объекты и служба присваивания имен OMG

Все служебные объекты (значения которым присваивает либо система управления поставщика, либо OMS) имеют собственный интерфейс IDL, определенный в данной Рекомендации, и ссылку на объект, зарегистрированную службой присваивания имен OMG, как изображено на рисунке 1, с использованием операции интерфейса службы присваивания имен bind(). В дальнейшем расположение реализации объекта может быть найдено, при необходимости, с помощью операции службы присваивания имен resolve().

Имена объектов могут быть либо "хорошо известными" (это означает, что имя было документировано и согласовано перед использованием), либо связываемыми в момент использования через другой интерфейс. В случае данной Рекомендации все объекты обслуживания являются "хорошо известными". Имя хорошо известного объекта может быть задокументировано посредством графа присваивания имен. На рисунке 2 представлен пример графа присваивания имен для некоторых объектов обслуживания данной Рекомендации в соответствии с соглашениями о присвоении имен, описанными в Рекомендации МСЭ-Т Q.816. На диаграмме показан идентификатор (Id), за которым следует точка, а за ней – значение вида. В целях упрощения на рисунке показано только четыре объекта обслуживания. Имена другим объектам должны присваиваться аналогичным образом.

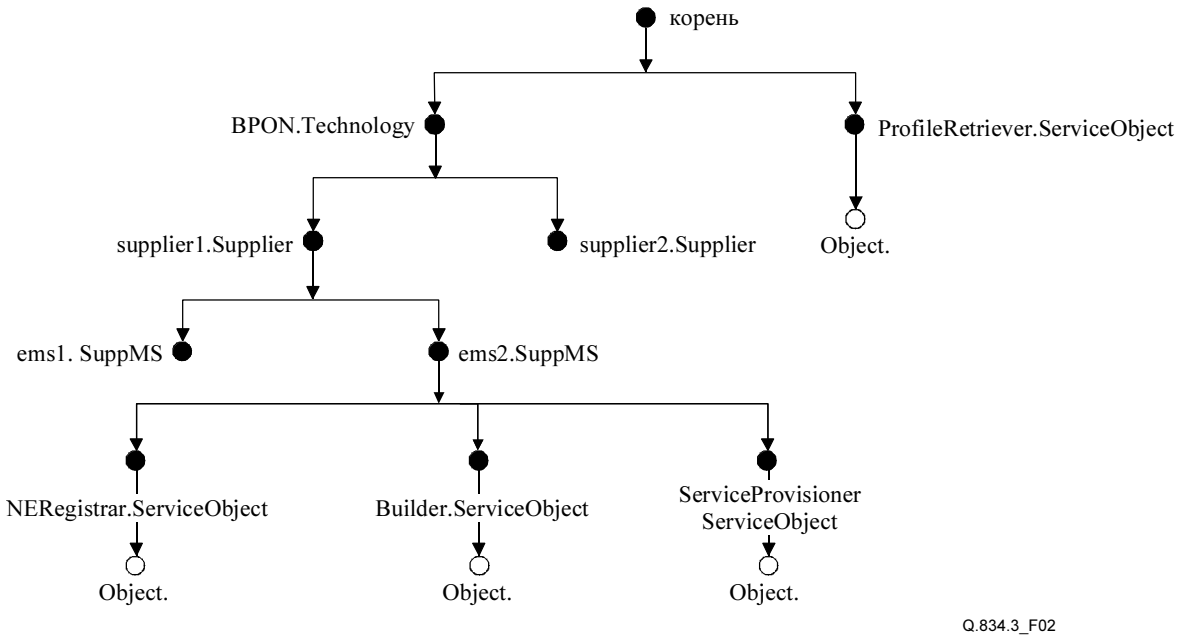


Рисунок 2/Q.834.4 – Граф присваивания имен

Для "хорошо известных" объектов требуется полный путь к имени. При запуске вновь установленной системы управления поставщика все реализации объектов обслуживания автоматически регистрируются в единой службе присваивания имен OMG, поддерживаемой оператором. Имя каждого объекта, зарегистрированного службой присваивания имен, имеет следующий синтаксис:

```
typedef     string      Istring;

struct NameComponent {
      Istring id;
      Istring kind;
};

typedef sequence<NameComponent> Name;
```

Одна из интерпретаций графа присваивания имен на рисунке 2 состоит в том, что поле "вид" в каждом из именуемых компонентов объектов обслуживания имеет значение пустой строки, а каждый

из показанных узлов представляет собой значение поля "id". Детали графа присваивания имен, так же как и его интерпретация, должны быть определены посредством соглашения между поставщиком и оператором и выходят за рамки данной Рекомендации.

7.2 Объекты домена

Система управления поставщика поддерживает большое количество управляемых объектов (или объектов домена)[2]. Эти объекты управляются косвенно через службы управления, предлагаемые системой управления поставщика. Однако для эффективного использования служб управления OMS часто требуются ссылки (идентификаторы) конкретных реализаций ресурсов. Когда идентификатор возвращается в качестве результата операции управления (или включается в уведомления), то предполагается, что он "известен" последующим службам управления, вызываемым из OMS. В этой спецификации идентификатор управляемого объекта имеет синтаксис, который позволяет системе управления поставщика информировать OMS по крайней мере одним из трех способов для взаимодействия с управляемыми объектами при последующих вызовах. Если управляемый объект доступен в IF1, то его имя CORBA присваивается, как описано в Рекомендации МСЭ-Т X.780. Если управляемый объект имеет поддерживаемый объект Façade, доступный в IF1, то имя CORBA объекта façade присваивается, как описано в Рекомендации МСЭ-Т X.780.1. Если нет ни одной из этих возможностей, то идентификатор состоит из ссылки, уникальной в пределах контекста управляемого домена системы управления поставщика. В общем случае детали этой ссылки основываются на специфической иерархии включений, определяемой при помощи соглашения между поставщиком и оператором, и выходят за пределы данной Рекомендации.

8 Организация файлов IDL

Спецификация интерфейса, предлагаемая в данной Рекомендации, состоит из родительского модуля, называемого "q834_4". Родительский модуль разделен на файлы IDL (более мелкие модули), соответствующие более мелким именуемым модулям. Каждый из этих файлов содержит одно или несколько определений интерфейса, представляющих конкретную службу управления, поддерживаемую в IF1. В таблице 1 приведен листинг, связывающий номер ссылки из Приложения C, имя файла IDL, содержащее интерфейсы IDL, и ссылку на соответствующее описание случаев использования из Рекомендации МСЭ-Т Q.834.3.

Таблица 1/Q.834.4 – Организация модуля q834_4

Ссылка на Приложение	Файлы IDL	Интерфейс IDL	Заголовок случая использования
C.1	Q834AccessControl	AccessControlMgr	Управление доступом
C.2	Q834Build	Builder	Сборка ресурсов BPON
C.3	Q834Common	ProbableCause MonitoringParameter RecordSetType PMCategory PhysicalLayerLoopback	
C.4	Q834ControlArchive	RecordSetMgr	Архивирование управления
C.5	Q834SoftwareDownload	DownloadMgr	Распределение программ

[2] Если система управления поставщика управляет 100–200 системами BPON (желательное требование размерности) и полностью поддерживает разделяемую базу знаний управления, содержащуюся в Рекомендациях МСЭ-Т Q.843.1 и Q.834.2, то количество реализаций управляемых объектов, поддерживаемых системой управления поставщика, варьируется от десятков миллионов до миллиардов в зависимости от предлагаемых услуг.

Таблица 1/Q.834.4 – Организация модуля q834_4

Ссылка на Приложение	Файлы IDL	Интерфейс IDL	Заголовок случая использования
		VersionRepository	Управление версиями программ NE
C.6	Q834EventPublisher	AlarmEventSupplier SecurityEventSupplier DiscoveryEventSupplier	Публикация событий
C.7	Q834MIBTransfer	MIBMover	Восстановление NE
C.8	Q834PerformanceManager	ImpairmentPersistence	RCAA & RCIA
		ReportController	Управление отчетами по мониторингу работы и трафика
C.9	Q834ProfileManager	ProfileConsumer ProfileUsageMgr ProfileRetriever	Управление профилями объектов
C.10	Q834Registrar	NERegistrar	Поддержка установленных ресурсов BPON
C.11	Q834ResourceAllocation	ResourceAllocator	Резервирование ресурсов
C.12	Q834SchedulerManagement	SchedulerMgr	Планировщик
C.13	Q834ServiceProvisioning	ServiceProvisioner	Служба обеспечения
C.14	Q834Synchroniser	Synchroniser	Выдача суммарных листингов текущих событий
			Синхронизация NE
C.15	Q834Test	TestActionPerformer	Тестирование
C.16	Q834FileTransfer	TransferMgr	Массовая передача

9 Модули

9.1 Модуль AccessControl

Данный модуль описывает функциональность создания, удаления, присваивания и использования информации управления доступом операторами в системе управления поставщика с использованием клиентских приложений с графическим интерфейсом GUI. В данном модуле пользователи могут быть распределены по группам. Разрешения могут даваться отдельным пользователям или группам пользователей. Пользователь обладает разрешениями всех групп, в которых он или она состоит. Каждое индивидуально выданное конкретному пользователю разрешение (см. 9.1.1.14 и 9.1.1.15) имеет преимущество над групповым. При групповых присваиваниях наивысший уровень разрешений действует для всех пользователей, включенных в группу. Предполагается, что привилегированный пользователь не запрашивает неоднозначных действий при создании или изменении групп пользователей или при присвоении разрешений индивидуальным пользователям.

Возможные действия определяются во всем модуле, как показано в таблице 2:

Таблица 2/Q.834.4 – Детали выполняемой деятельности

Имя поля	Определение	Синтаксис	Комментарии
activityLevel	Определяет уровень доступа для деятельности	enum	MonitorOnly allowedToExecute noAccess
activityType	Определяет тип деятельности	short	Определяется в форме различных констант в интерфейсе AccessControlMgr
administrationDomainList	Идентификатор, выдаваемый OMS или оператором в процессе регистрации для указания административного домена, которому принадлежит элемент сети	UserLabel	

9.1.1 Интерфейс AccessControlMgr

9.1.1.1 setPasswordPolicy (установка политики паролей)

Данная операция позволяет привилегированным пользователям управлять политикой паролей.

Сигнатура операции **setPasswordPolicy** приведена ниже:

```
void setPasswordPolicy (in PasswordPolicyType passwordPolicy)
                        raises (AccessDenied);
```

Входной параметр **passwordPolicy** определяет политику паролей, которая применяется системой управления поставщика и состоит из политики входов пользователей UserLoginPolicy и политики сессии SessionPolicy. UserLoginPolicy диктует следующие параметры, относящиеся к идентификатору пользователя userId и паролю: минимальный размер userId, минимальный размер пароля пользователя, длина интервала в днях до момента, когда пароль может быть повторно использован, максимальное количество разрешенных попыток входа до того, как доступ пользователя будет блокирован на день, время действия пароля (в днях), должен ли пароль содержать алфавитно-цифровую смесь, должен ли в нем содержаться хотя бы один специальный символ, могут или нет в пароле содержаться повторяющиеся символы и может ли идентификатор пользователя быть частью пароля.

SessionPolicy определяет следующие параметры, относящиеся к сессии: длительность периода времени, в течение которого сессия может быть неактивной до того, как будет прервана, период времени, после которого неактивный пользователь отключается, и максимальное количество сессий, разрешенных для одного идентификатора пользователя.

Поскольку в системе управления поставщика может существовать только одна passwordPolicy, нет необходимости иметь метод ее создания. Либо в такой системе существует политика по умолчанию, либо первый метод, устанавливающий ее значение, может одновременно и создавать ее.

Возвращаемое значение имеет тип **void**.

9.1.1.2 passwordPolicyGet (получить политику паролей)

Данная операция позволяет привилегированным пользователям получать настройки политики, относящиеся к синтаксису данных обмена и значения таймеров, используемые при входе в систему управления поставщика.

Сигнатура операции **passwordPolicyGet** приведена ниже:

```
PasswordPolicyType passwordPolicyGet()
                   raises (AccessDenied);
```

Для данной операции не требуется входных параметров.

Возвращаемое значение имеет тип **PasswordPolicyType** и содержит подробную регистрацию действий при входе в систему управления поставщика.

9.1.1.3 userListGet

Данная операция позволяет привилегированным пользователям получить список идентификаторов пользователя (ids), имеющих какую-либо форму доступа к системе управления поставщика, а также сведения о разрешенных им видах деятельности и принадлежности к группам.

Сигнатура операции **userListGet** приведена ниже:

```
UserSeqType userListGet ()
                    raises (AccessDenied);
```

Для этой операции не требуется входных параметров.

Возвращаемое значение имеет тип **UserSeqType** и представляет собой список идентификаторов пользователя (ids), имеющих какую-либо форму доступа к системе управления поставщика, а также сведения о разрешенных им видах деятельности и принадлежности к группам.

9.1.1.4 userGroupListGet

Данная операция позволяет привилегированным пользователям получить сведения о группах пользователей, имеющих доступ к системе управления поставщика, вместе со списками членов групп и данными о разрешенных им видах деятельности.

Сигнатура операции **userGroupListGet** приведена ниже:

```
UserGroupSeqType userGroupListGet ()
                    raises (AccessDenied);
```

Для этой операции не требуется входных параметров.

Возвращаемое значение имеет тип **UserGroupSeqType** и содержит сведения о группах пользователей, имеющих доступ к системе управления поставщика, вместе со списками членов групп и данными о разрешенных им видах деятельности.

9.1.1.5 userGet

Данная операция позволяет привилегированным пользователям получить данные о членстве пользователя в группе и о разрешенных ему видах деятельности.

Сигнатура операции **userGet** приведена ниже:

```
UserType userGet (
          in UserIdType userId )
          raises (AccessDenied, UnknownUserIds);
```

Входной параметр **userId** идентифицирует запрашиваемого пользователя.

Возвращаемое значение имеет тип **UserType** и содержит данные о членстве пользователя в группе и о разрешенных ему видах деятельности.

9.1.1.6 userGroupGet

Данная операция позволяет привилегированным пользователям получить данные о пользователях – членах группы и разрешенных им видах деятельности.

Сигнатура операции **userGroupGet** приведена ниже:

```
UserGroupType userGroupGet (
                    in UserLabelType userGroupId)
                    raises (AccessDenied, UnknownUserGroupId);
```

Входной параметр **userGroupId** идентифицирует группу пользователей.

Возвращаемое значение имеет тип **UserGroupType** и содержит идентификатор группы пользователей, идентификаторы пользователей – членов группы и сведения о разрешенных им видах деятельности.

9.1.1.7 createUserGroup

Данная операция позволяет привилегированным пользователям создать новую группу пользователей. Группы пользователей используются для предоставления определенным пользователям определенных видов доступа к определенным элементам сети. Привилегированный пользователь создает группу пользователей вместе со списком разрешенных для нее видов деятельности.

Сигнатура операции **createUserGroup** приведена ниже:

```
void createUserGroup (
                    in UserLabelType userGroupId,
                    in TargetActivitySeqType targetAdditions)
                    raises (DuplicateUserGroupId, UnknownTargets, AccessDenied);
```

Входной параметр **userGroupId** идентифицирует создаваемую группу. Параметр UserGroupId не может иметь значения пустой строки. Входной параметр **targetAdditions** идентифицирует виды деятельности (TargetActivities), разрешенные пользователям, принадлежащим к группе.

Возвращаемое значение имеет тип **void**.

9.1.1.8 modifyUserGroup

Данная операция позволяет привилегированным пользователям добавлять или удалять виды деятельности членам группы. Привилегированный пользователь указывает группу и список видов добавляемых или удаляемых видов деятельности. Система управления поставщика сначала обрабатывает удаления, а затем добавления. Это позволяет, в частности, повышать уровень разрешения для конкретных видов деятельности.

Сигнатура операции **modifyUserGroup** приведена ниже:

```
TargetActivitySeqType modifyUserGroup (
                        in UserLabelType userGroupId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserGroupId, UnknownTargets,
                        AccessDenied);
```

Входной параметр **userGroupId** определяет группу пользователей, которой оператор хочет добавить или удалить определенные виды деятельности. Входной параметр **targetAdditions** определяет виды деятельности (TargetActivities), которые нужно добавить группе. Входной параметр **targetDeletions** определяет TargetActivities, которые следует отменить для группы.

Возвращаемое значение имеет тип **TargetActivitySeqType** и содержит обновленный список видов деятельности (TargetActivities), которые разрешены пользователям, принадлежащим к данной группе.

9.1.1.9 deleteUserGroup

Данная операция позволяет привилегированным пользователям удалять группу пользователей. Удаление завершается успешно, только если группа пуста.

Сигнатура операции **deleteUserGroup** поканаза ниже:

```
void deleteUserGroup (
               in UserLabelType userGroupId)
               raises (AccessDenied, UserGroupNotEmpty, UnknownUserGroupId);
```

Входной параметр **userGroupId** определяет удаляемую группу.

Возвращаемое значение имеет тип **void**.

9.1.1.10 addUsersToGroup

Данная операция позволяет привилегированному пользователю добавлять новых пользователей к существующей группе.

Сигнатура операции **addUsersToGroup** приведена ниже:

```
void addUsersToGroup (
               in UserLabelType userGroupId,
               in UserIdSeqType userIdList)
               raises (AccessDenied, UnknownUserGroupId);
```

Входной параметр **userGroupId** определяет группу, к которой должны быть добавлены новые пользователи. Входной параметр **userIdList** определяет набор userId, которые должны быть добавлены к существующей группе пользователей.

Возвращаемое значение имеет тип **void**.

9.1.1.11 deleteUsersFromGroup

Данная операция позволяет привилегированным пользователям удалять пользователей из группы.

Сигнатура операции **deleteUsersFromGroup** приведена ниже:

```
void deleteUsersFromGroup (
              in UserLabelType userGroupId,
              in UserIdSeqType userIdList)
              raises (AccessDenied, UnknownUserGroupId, UnknownUserIds );
```

Входной параметр **userGroupId** определяет группу, из которой нужно удалить пользователей. Входной параметр **userIdList** определяет набор userId, которые должны быть удалены из группы.

Возвращаемое значение имеет тип **void**.

9.1.1.12 getPermissionList

Данная операция позволяет привилегированным пользователям получить список видов деятельности, разрешенных указанному пользователю.

Сигнатура операции **getPermissionList** приведена ниже:

```
TargetActivitySeqType getPermissionList (
                        in UserIdType userId)
                        raises (UnknownUserIds, AccessDenied);
```

Входной параметр **userId** определяет пользователя, чей список TargetActivities хочет получить привилегированный пользователь.

Возвращаемое значение имеет тип **TargetActivitySeqType** и представляет собой список TargetActivities, разрешенных указанному пользователю.

9.1.1.13 modifyPermissionList

Данная операция позволяет привилегированным пользователям изменять список видов деятельности, разрешенных пользователю. Привилегированный пользователь указывает userId и список видов деятельности, которые должны быть добавлены или удалены. Система управления поставщика сначала обрабатывает удаления, а затем добавления. Это позволяет, например, повышать уровень разрешений для данного вида деятельности.

Сигнатура операции **modifyPermissionList** приведена ниже:

```
TargetActivitySeqType modifyPermissionList (
                        in UserIdType userId,
                        in TargetActivitySeqType targetAdditions,
                        in TargetActivitySeqType targetDeletions)
                        raises (UnknownUserIds, UnknownTargets, AccessDenied);
```

Входной параметр **userId** определяет пользователя, чьи разрешенные виды деятельности привилегированный пользователь хочет изменить. Входной параметр **targetAdditions** определяет разрешенные виды деятельности, которые должны быть добавлены указанному пользователю. Входной параметр **targetDeletions** определяет разрешенные виды деятельности, которые должны быть удалены у указанного пользователя.

Если пользователь принадлежит нескольким группам, имеющим те же разрешенные виды деятельности, то уровень доступа для него устанавливается как наивысший из всех групп. Когда пользователь и группа, к которой он принадлежит, относятся к одному и тому же административному домену, то уровень доступа (activityLevel) пользователя имеет преимущество перед activityLevel группы.

Возвращаемое значение **TargetActivitySeqType** представляет собой список видов деятельности, разрешенных данному пользователю после модификации.

9.1.1.14 createUser

Данна операция позволяет привилегированным пользователям создавать нового пользователя. Привилегированный пользователь создает новый userId, пароль для него и список видов деятельности, разрешенных данному пользователю.

Сигнатура операции **createUser** приведена ниже:

```
void createUser (
            in UserIdType userId,
            in PasswordType password,
            in TargetActivitySeqType targetAdditions)
            raises (DuplicateUserId, UnknownTargets, AccessDenied,
            UserLoginPolicyViolation);
```

Входной параметр **userId** определяет идентификационную информацию, которая будет связана с новым пользователем. Входной параметр **password** определяет пароль, который будет связан с новым идентификатором пользователя. Входной параметр t**argetAdditions** определяет виды деятельности (TargetActivities), разрешенные данному пользователю.

Возвращаемое значение имеет тип **void**.

9.1.1.15 deleteUser

Данная операция позволяет привилегированному пользователю удалять существующего пользователя.

Сигнатура операции **deleteUser** приведена ниже:

```
void deleteUser (
            in UserIdType userId)
            raises (UnknownUserIds, AccessDenied);
```

Входной параметр **userId** определяет удаляемого пользователя.

Возвращаемое значение имеет тип **void**.

9.1.1.16 resetPassword

Данная операция позволяет привилегированным пользователям переустанавливать пароль заданному пользователю. Она имеет место тогда, когда старый пароль недоступен, и привилегированный пользователь должен установить для пользователя новый пароль. При первом входе система управления поставщика предлагает пользователю сменить пароль.

Сигнатура операции **resetPassword** приведена ниже:

```
void resetPassword (
              in UserIdType userId,
              in PasswordType newPassword)
              raises (UnknownUserIds, UserLoginPolicyViolation, AccessDenied);
```

Входной параметр **userId** определяет пользователя, пароль которого должен быть удален. Входной параметр **newPassword** определяет пароль, который должен быть связан с userId.

Возвращаемое значение имеет тип **void**.

9.1.1.17 Исключительные ситуации

Исключительная ситуация **AccessDenied** (доступ запрещен) возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **DuplicateUserId** (повторяющийся идентификатор пользователя) возникает, если userId, указанный при создании нового пользователя, уже существует в базе данных системы управления поставщика.

Исключительная ситуация **DuplicateUserGroupId** (повторяющийся идентификатор группы пользователей) возникает, если userGroupId, указанный при создании новой группы пользователей,

уже существует в системе управления поставщика. Другими словами, политика системы управления поставщика определяет порядок присваивания меток пользователя (User Label) в идентификаторах групп пользователей для всех пользователей, опознаваемых ею.

Исключительная ситуация **UnknownTargets** (неизвестные действия) возникает, если указанный список TargetActivities не может быть идентифицирован.

Исключительная ситуация **UnknownUserGroupId** (неизвестный идентификатор группы) возникает, если указанный UserGroupId не может быть идентифицирован.

Исключительная ситуация **UnknownUserIds** (неизвестный идентификатор пользователя) возникает, если какой-либо userId не может быть идентифицирован.

Исключительная ситуация **UserGroupNotEmpty** (группа не пуста) возникает, если имеет место попытка удалить не пустую группу пользователей (в группе есть зарегистрированные пользователи).

Исключительная ситуация **UserLoginPolicyViolation** (нарушение политики входа) возникает, если пароль, заданный для нового пользователя или при смене пароля, не соответствует принципам политики входа в систему (UserLoginPolicy).

9.2 Модуль сборки

Система управления поставщика строит свою модель управления по запросам от OMS или оператора в зависимости от планируемого оборудования как часть деятельности, предшествующей поставке. Необходимые для этого ресурсы включают в себя узлы (OLT, ONT, ONU, NT) и добавляемые модули. Если оборудование уже установлено, то для завершения поставки установленных ресурсов используются операции модификации. При этом защитные группы устанавливаются как для уже готовых, так и для подготавливаемых к поставке ресурсов.

9.2.1 Интерфейс Builder

9.2.1.1 buildNode

Данная операция строит элемент сети (NE) в системе управления поставщика. В результате ее в информационной модели управления, поддерживаемой системой управления поставщика, автоматически создается набор управляемых объектов. В зависимости от конкретной реализации оборудования, полученного от поставщика, реализации установок equipmentHolderF для стоек и слотов, установок NEFSAN и установок physicalPathTPF для интегрированных портов служат примером того, что может быть автоматически создано.

Если NE представляет собой ONT или ONU, то в результате данной операции из имеющегося системного диапазона автоматически удаляется диапазон, необходимый для поддержки встраиваемых операций канала между OLT и ONT или ONU.

Сигнатура операции **buildNode** приведена ниже:

```
ManagedEntityIdType buildNode (
                    in NEKindType nEKind,
                    in string supplierName,
                    in string location,
                    in VersionType version,
                    in SerialNumType serialNum,
                    in NameSeqType alarmSeverityProfiles,
                    in NameSeqType thresholdDataProfiles,
                    in SlotAssignmentSeqType slotAssignmentList,
                    in ManagedEntityIdType port,
                    in string modelCode,
                    in string systemTitle,
                    in VersionSeqType softwareVersions,
                    in UserLabelType nEUserLabel,
                    in ExternalTimeType externalTime,
                    in SystemTimingType systemTiming,
                    in AdministrationDomainType administrationDomain)
                    raises (UnrecognisedVersion, InvalidSerialNumSyntax,
                    DuplicateSerialNumber, UnknownProfiles,
                    UnknownManagedEntity, DuplicateUserLabel, AccessDenied,
```

```
                InvalidExternalTime, UnknownSystemTimingSource,
                ProfileSuspended);
```

Входной параметр **nEKind** определяет тип собираемого NE. Возможные типы NE: OLT, ONT, ONU или NT. Входные параметры **supplierName** и **version** определяют поставщика NE и версию устанавливаемого оборудования. Входной параметр **location** обозначает физическое расположение собираемого NE. Входной параметр **serialNum** создает уникальную строку, соответствующую NE. Входной параметр **alarmSeverityProfiles** определяет профили для конфигурирования кодов серьезности индивидуальных сигналов тревоги, которые будут выдаваться NE. Входной параметр **thresholdDataProfiles** определяет профили, которые будут использоваться при конфигурировании пороговых значений для генерации предупреждений о достижении пороговых величин (TCA) на NE. Входной параметр **thresholdDataProfiles** определяет профили, используемые при настройке пороговых значений для генерации TCA на NE. Входной параметр **slotAssignmentList** определяет подходящие способы присоединения подключаемых модулей к слотам NE. Если номер слота не встречается в списке, то предполагается, что для него нет никаких конкретных рекомендаций и, следовательно, нет ограничений на тип карты, помещаемой в этот слот. То же самое верно и для случая, когда подключаемому модулю (plugIn) соответствует пустая строка. Входной параметр **port** определяет порт PON в OLT, если конструируемый NE имеет тип либо ONT, либо ONU. Параметр **port** определяет порт ONU, обслуживающий NT, находящийся в процессе сборки. В случае сборки OLT значение параметра **port** = null. Входной параметр **modelCode** представляет собой уникальную строку, идентифицирующую тип сетевого ресурса. Этот входной параметр в первую очередь представляет интерес в случае подготовки к созданию ONT или NT. Входной параметр **systemTitle** определяет задаваемую оператором метку, которая будет присвоена узлу. Входной параметр **softwareVersions** определяет версию программного обеспечения, которое будет использоваться на узле. Входной параметр **nEUserLabel** задает уникальную маркировку оператора для собираемого NE. Входной параметр **externalTime** устанавливает для NE текущее время в обобщенной форме. Входной параметр **systemTiming** определяет для NE источник входных сигналов времени, которые будут использоваться при временной синхронизации. Если от OMS поступает не указанное значение параметра **systemTiming** типа "enum", то NE будет использовать источник сигналов времени, установленный по умолчанию. Входной параметр **administrationDomain** определяет домен, которому принадлежит NE.

Возвращаемое значение типа **ManagedEntityIdType** задает идентификатор нового NE, созданного в результате операции.

9.2.1.2 assignUserLabelsToNE

Данная операция присваивает NE административную метку оператора. Она требуется, когда OMS первый раз изучает NE с помощью автоматического обнаружения.

Сигнатура операции **assignUserLabelsToNE** приведена ниже:

```
void assignUserLabelsToNE (
               in SerialNumType serialNum,
               in UserLabelType nEUserLabel,
               in AdministrationDomainType administrationDomain)
               raises (InvalidSerialNumSyntax, DuplicateSerialNumber,
               DuplicateUserLabel, AccessDenied);
```

Входной параметр **serialNum** идентифицирует указанный NE. Параметр **nEUserLabel** задает для NE метку, присваиваемую оператором. Параметр **administrationDomain** идентифицирует домен, к которому присоединен NE.

Возвращаемое значение имеет тип **void**.

9.2.1.3 modifyNode

Данная операция инициирует реконфигурацию и обновление указанных параметров, связанных с NE.

Сигнатура операции **modifyNode** приведена ниже:

```
void modifyNode (
            in ManagedEntityIdType managedEntityId,
            in SlotAssignmentSeqType newSlotAssignmentList,
            in NameSeqType newAlarmSeverityProfiles,
            in NameSeqType newThresholdDataProfiles,
            in ManagedEntityIdType port,
            in string newModelCode,
            in UserLabelType newNEUserLabel,
            in ExternalTimeType externalTime,
            in AdministrationDomainType administrationDomain)
            raises(UnknownManagedEntity, UnknownNE, InvalidSlotAssignmentList,
            UnknownProfiles, DuplicateUserLabel, AccessDenied,
            InvalidExternalTime, ProfileSuspended );
```

Входной параметр **managedEntityId** идентифицирует NE, который будет модифицироваться. Параметр **newSlotAssignmentList** идентифицирует новое присвоение допустимых подключаемых модулей слотам NE. Если номер слота не упоминается в списке, то предполагается, что для этого слота изменений нет. Если значение поля plugIn представляет сбой пустую строку, то это значит, что нет ограничений для типов карт, которые могут быть помещены в слот, и если присоединенный модуль в дальнейшем удаляется из слота, то никаких сигналов тревоги не возникает. Параметр **alarmSeverityProfiles** определяет профили, с помощью которых конфигурируется серьезность индивидуальность сигналов тревоги, которые будет выдавать NE. Параметр **thresholdDataProfiles** определяет профили, которые будут использоваться при конфигурировании пороговых значений генерации TCA на NE. Параметр **port** определяет порт PON на OLT, если есть изменение во взаимодействии между планируемой ONT или ONU и портом OLT. Параметр **port** определяет порт ONU если есть изменение во взаимодействии между планируемым NT и портом ONU. Параметр **newModelCode** выдает уникальную строку, идентифицирующую тип сетевого ресурса. Этот входной параметр интересен в первую очередь тогда, когда модифицируемый узел является узлом ONT или NT. Параметр **newNEUserLabel** обеспечивает новую метку пользователя, которая будет применена к NE. Параметр **externalTime** задает новое эталонное время для NE. Параметр **administrationDomain** определяет домен, к которому в этот раз присоединяется NE.

Возвращаемое значение имеет тип **void**.

9.2.1.4 deleteNode

Данная операция удаляет NE из систему управления поставщика и прекращает предыдущие необработанные запросы. В результате этой операции все управляемые элементы, автоматически созданные соответствующей операцией buildNode также удаляются. Если любые относящиеся к нему управляемые объекты продолжают подключаться к услугам, то возникает исключительная ситуация **RemainingContainedManagedEntities**. Диапазон, зарезервированный для данного узла, также удаляется.

Сигнатура операции **deleteNode** приведена ниже:

```
void deleteNode (
            in ManagedEntityIdType managedEntityId)
            raises (UnknownNE, RemainingContainedManagedEntities, AccessDenied,
            RemainingReservations, RemainingSubnetworkConnections);
```

Входной параметр **managedEntityId** определяет NE, который будет удален.

Возвращаемое значение имеет тип **void**.

9.2.1.5 modifyPort

Данная операция модифицирует параметры порта, идентифицируемого посредством **physicalPathTPId**.

Сигнатура операции **modifyPort** приведена ниже:

```
void      modifyPort     (
               in ManagedEntityIdType physicalPathTPId,
               in NameSeqType newAlarmSeverityProfiles,
               in NameSeqType newThresholdDataProfiles,
               in NameSeqType newPortProfiles,
               in string newFrameFormat,
               in AdministrativeStateType administrativeState,
               in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
               in LoopbackLocationIdSeqType newLoopbackLocationIds,
               in unsigned long newInterfaceSpeed,
               in unsigned long aRCTimer)
               raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
               InterfaceSpeedNotChangeable, ProfileSuspended );
```

Входной параметр **physicalPathTPId** определяет порт, который будет модифицироваться. Входной параметр **alarmSeverityProfiles** определяет профили для конфигурирования уровней серьезности сигналов тревоги, о которых будет сообщать NE. Входной параметр **thresholdDataProfiles** определяет профили, используемые при конфигурировании пороговых значений для генерации TCA на NE. Входной параметр **newPortProfiles** определяет профили, используемые для завершения настроек порта. Примеры таких профилей включают следующие (но не ограничиваются ими): ATMNetworkAccessProfile, UNIInfo, EthernetProfile, CESServiceProfile, MACBridgeServiceProfile, LESServiceProfile, AAL1Profile и AAL5Profile. Параметр **newFrameFormat** определяет новый формат кадра, который будет завершен и сгенерирован для физического порта, если формат кадра допускает конфигурирование. Параметр **administrativeState** задает модифицированные установки для данного параметра. Параметр **newLoopbackLocationIds** определяет новые идентификаторы расположения для кольцевой схемы, связанной с физическим портом. Параметр **newInterfaceSpeed** определяет новую скорость интерфейса физического порта, если она может настраиваться. Параметр **aRCTimer** задает неотрицательное время (в секундах), в течение которого сетевой ресурс первоначально определяет допустимый сигнал перед тем, как выдать в порт любой коммуникационный сигнал тревоги.

Возвращаемое значение имеет тип **void**.

9.2.1.6 buildPlugInUnit

Данная операция создает в системе управления поставщика подключаемый модуль в процессе выполнения предварительных действий. В результате этой операции в информационной модели управления, реализуемой системой, автоматически создается набор управляемых элементов. В зависимости от типа подключаемого элемента автоматически создаются различные точки завершения. Присвоение слота узлу автоматически обновляется, чтобы включить изменения, запрошенные данной операцией.

Сигнатура операции **buildPlugInUnit** приведена ниже:

```
ManagedEntityIdType buildPlugInUnit (
                        in ManagedEntityIdType nEId,
                        in NameType alarmSeverityProfile,
                        in UserLabelType plugInUnitUserLabel,
                        in string modelCode,
                        in AdministrativeStateType administrativeState,
                        in ManagedEntityIdType equipmentHolder)
                        raises (UnknownNE, DuplicateUserLabel,
                        AccessDenied, UnknownManagedEntity,
                        InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                        InvalidSlotAssignmentList, UnknownProfiles,
                        ProfileSuspended );
```

Входной параметр **nEId** обеспечивает уникальное имя NE, который содержит подключаемый модуль. Параметр **plugInUnitUserLabel** выдает идентификатор подключаемого модуля. Параметр **alarmSeverityProfile** определяет профиль для конфигурирования уровней сигналов тревоги при отказах оборудования, относящихся к подключаемому модулю. Параметр **modelCode** определяет тип подключаемого модуля. Параметр **administrativeState** задает начальные установки для данного

параметра. Параметр **equipmentHolder** определяет расположение слота, в который будет установлен подключаемый модуль.

Возвращаемое значение типа **ManagedEntityIdType** содержит уникальный идентификатор для конструируемой схемы.

9.2.1.7 modifyPlugInUnit

Данная операция модифицирует атрибуты подключаемого модуля в системе управления поставщика.

Сигнатура операции **modifyPlugInUnit** приведена ниже:

```
ManagedEntityIdType modifyPlugInUnit (
                    in ManagedEntityIdType plugInUnitId,
                    in NameType alarmSeverityProfile newAlarmSeverityProfile,
                    in string newModelCode,
                    in ManagedEntityIdType newEquipmentHolder,
                    in UserLabelType newPlugInUnitUserLabel,
                    in AdministrativeStateType newAdministrativeState)
                    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                    InvalidEquipmentCode, SlotAlreadyAssigned, UnknownSlot,
                    InvalidSlotAssignmentList, InvalidUserLabelSyntax,
                    ProfileSuspended);
```

Входной параметр **plugInUnitId** определяет модифицируемый управляемый объект. Параметр **newAlarmSeverityProfile** меняет профиль конфигурирования уровней сигналов тревоги, возникающих при отказах оборудования подключаемого модуля. Параметр **newModelCode** изменяет тип подключаемого модуля. Параметр **newEquipmentHolder** меняет слот, предназначенный для подключаемого модуля. Параметр **newPlugInUserLabel** выдает новую метку пользователя для подключаемого модуля. Параметр **newAdministrativeState** задает модифицированные установки данного параметра.

Возвращаемое значение имеет тип **ManagedEntityIdType** и выдает уникальный идентификатор для модифицируемой схемы.

9.2.1.8 deletePlugInUnit

Данная операция удаляет подключаемый модуль из системы управления поставщика. В результате ее выполнения все управляемые объекты, автоматически созданные соответствующей операцией buildPlugInUnit, также удаляются. Если какие-либо их связи остались, то возникает исключительная ситуация **RemainingSubnetworkConnections**. Она предназначена для удаления предварительной информации, которая более не требуется оператору.

Сигнатура операции **deletePlugInUnit** приведена ниже:

```
void deletePlugInUnit (
                    in ManagedEntityIdType plugInUnitId)
                    raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                    AccessDenied, RemainingReservations);
```

Входной параметр **plugInUnitId** определяет подключаемый модуль, который будет удален.

Возвращаемое значение имеет тип **void**.

9.2.1.9 buildProtectionGrouping

Данная операция создает защищенные группы (protectionGrouping) в системе управления поставщика. Ни один порт не может принадлежать более чем одной защищенной группе и должен быть определен до выполнения этой операции. За пределами диапазона IF1 поставщик и оператор приходят к общему пониманию систематизации средств защиты, поддерживаемых оборудованием поставщика. Если в списке protectionUnitList содержится защищенный порт, то в системе должен быть также по крайней мере один защищающий порт. Все порты должны иметь одинаковые характеристики физического пути.

Сигнатура операции **buildProtectionGrouping** приведена ниже:

```
ManagedEntityIdType buildProtectionGrouping (
                        in ProtectionParameterType protectionParameters,
                        in ProtectionUnitSeqType protectionUnitList)
                        raises (InvalidProtectionScheme, AccessDenied );
```

Входной параметр **protectionParameters** задает характеристики схемы защиты. Параметр **protectionUnitList** определяет защищающий и защищаемый порты сетевого ресурса.

Возвращаемое значение имеет тип **ManagedEntityIdType** и выдает уникальный идентификатор для взаимодействия между защищаемым и защищающим портами.

9.2.1.10 modifyProtectionParameters

Данная операция модифицирует схему защиты для идентифицируемой защищаемой группы. Если либо m (= количеству защищающих портов), либо n (= количеству защищаемых портов) уменьшаются, то для OMS необходимо предусмотреть вызов операции modifyProtectionUnitList, чтобы избежать возникновения исключительной ситуации нарушения схемы защиты. Если m или n увеличивается, то modifyProtectionUnitList вызывать не нужно.

Сигнатура операции **modifyProtectionParameters** приведена ниже:

```
void modifyProtectionParameters (
                      in ManagedEntityIdType protectionGroupingId,
                      in ProtectionParameterType newProtectionParameters)
                      raises (UnknownManagedEntity, InvalidProtectionScheme,
                      AccessDenied);
```

Входной параметр **protectionGroupingId** определяет управляемый объект, который будет модифицироваться данной операцией. Параметр **newProtectionParameters** обеспечивает замену существующих параметров защиты.

Возвращаемое значение имеет тип **void**.

9.2.1.11 modifyProtectionUnitList

Данная операция добавляет в список или удаляет из списка защищаемых или защищающих портов указанную группу защиты. Удаление всех защищающих портов может иметь место только в случае, если все защищаемые порты также удалены. Добавление защищаемых или защищающих портов не может превысить значений m или n, заданных в рамках структуры данных, создаваемой операцией protectionParameters, относящейся к группе защиты. Ни один порт не может принадлежать более чем одной группе защиты и должен быть установлен до выполнения данной операции. Все порты должны иметь одинаковые характеристики физического пути.

Сигнатура операции **modifyProtectionUnitList** приведена ниже:

```
void modifyProtectionUnitList (
                     in ManagedEntityIdType protectionGroupingId,
                     in ProtectionUnitSeqType deltaProtectionUnitList,
                     in boolean addDeleteInd)
                     raises (UnknownManagedEntity, InvalidProtectionScheme,
                     AccessDenied);
```

Входной параметр **protectionGroupingId** определяет управляемый объект, модифицируемый данной операцией. Параметр **deltaProtectionUnitList** задает изменения защищающих и защищаемых портов. Входной параметр **addDeleteInd** показывает, является ли изменение добавлением или удалением.

Возвращаемое значение имеет тип **void**.

9.2.1.12 deleteProtectionGrouping

Данная операция удаляет группу защиты портов из системы управления поставщика. В результате выполнения этой операции все управляемые объекты, автоматически созданные соответствующей операцией buildProtectionGrouping, также удаляются.

Сигнатура операции **deleteProtectionGrouping** приведена ниже:

```
void deleteProtectionGrouping (
                    in ManagedEntityIdType protectionGroupingId)
                    raises (UnknownManagedEntity, AccessDenied);
```

Входной параметр **protectionGroupingId** определяет группу защиты, которая будет удалена.

Возвращаемое значение имеет тип **void**.

9.2.1.13 buildBridge

Данная операция создает в системе управления поставщика мост MAC. Все порты должны быть установлены перед этой операцией. Все порты UNI должны быть портами LAN. Эта операция не требуется, если все порты UNI LAN элемента сети автоматически принадлежат одному порту и установки параметров моста по умолчанию устраивают оператора.

Сигнатура операции **buildBridge** приведена ниже:

```
ManagedEntityIdType buildBridge (
                        in NameType mACBridgeProfile,
                        in ManagedEntityIdType uplinkPort,
                        in ManagedEntityIdSeqType uNIPortList)
                        raises (UnknownProfiles, AccessDenied,
                        UnknownManagedEntity, ProfileSuspended);
```

Входной параметр **mACBridgeProfile** задает характеристики моста согласно ANSI/IEEE 802.1D. Параметр **uplinkPort** определяет физический порт OLT, взаимодействующий по интерфейсу с магистральной сетью уровня IP . Этот параметр имеет значение пустой строки, если в OLT нет такого интерфейса. Параметр **uNIPortList** определяет физические порты UNI, относящиеся к мосту.

Возвращаемое значение типа **ManagedEntityIdType** выдает уникальный идентификатор конструируемого моста.

9.2.1.14 modifyBridgeProfile

Данная операция изменяет характеристики функции передачи по мосту путем изменения сервисного профиля MAC, относящегося к данному мосту.

Сигнатура операции **modifyBridgeProfile** приведена ниже:

```
void modifyBridgeProfile (
                    in ManagedEntityIdType bridgeId,
                    in NameType newMACBridgeProfile)
                    raises (UnknownManagedEntity, UnknownProfiles, AccessDenied,
                    ProfileSuspended);
```

Входной параметр **bridgeId** определяет управляемый объект, модифицируемый данной операцией. Параметр **newMACBridgeProfile** обеспечивает замену существующего профиля моста.

Возвращаемое значение имеет тип **void**.

9.2.1.15 modifyBridgePortList

Данная операция либо добавляет порт в список портов UNI, принадлежащих группе моста, либо удаляет его из этого списка. Возможно удаление всех портов UNI. Порт UNI не может быть удален, если существует связанное с ним активное подключение к службе. Если все порты UNI LAN должны принадлежать одному и тому же мосту, то данная операция не требуется.

Сигнатура операции **modifyBridgePortList** приведена ниже:

```
void modifyBridgePortList (
                  in ManagedEntityIdType bridgeId,
                  in ManagedEntityIdSeqType deltaUNIPortList,
                  in boolean addDeleteInd)
                  raises (UnknownManagedEntity, RemainingSubnetworkConnections,
                  AccessDenied);
```

Входной параметр **bridgeId** определяет управляемый объект, который будет модифицироваться данной операцией. Параметр **deltaUNIPortList** обеспечивает изменения в требуемых портах UNI. Входной параметр **addDeleteInd** показывает, является ли изменение добавлением или удалением.

Возвращаемое значение имеет тип **void**.

9.2.1.16 deleteBridge

Данная операция удаляет мост из системы управления поставщика. В результате ее выполнения также автоматически удаляются все управляемые объекты, автоматически созданные при выполнении соответствующей операции buildBridge. Мост нельзя удалить, если имеются активные внешние соединения с подсетью, связанной с ним.

Сигнатура операции **deleteBridge** приведена ниже:

```
void deleteBridge (
            in ManagedEntityIdType bridgeId)
            raises (UnknownManagedEntity, AccessDenied,
            RemainingSubnetworkConnections);
```

Входной параметр **bridgeId** определяет удаляемый мост.

Возвращаемое значение имеет тип **void**.

9.2.1.17 buildVPNetworkCTP

Данная операция создает в системе управления поставщика виртуальный путь (VP) к сетевой точке окончания соединения (CTP). Порт, который содержит VP к сетевой CTP, должен быть определен до выполнения этой операции. Эта операция используется для задания VP к сетевым CTP в целях поддержки установочного тестирования. Требуются как операции создания, так и удаления[3].

Сигнатура операции **buildVPNetworkCTP** приведена ниже:

```
ManagedEntityIdType buildVPNetworkCTP (
                        in ManagedEntityIdType port,
                        in short vPI,
                        in NameType trafficDescriptorProfileName,
                        in ATMOverbookingFactorType overbookingFactor,
                        in UserLabelType userLabel,
                        in SegmentEndpointIndType segmentEndpointInd)
                        raises (UnknownProfiles, AccessDenied,
                        UnknownManagedEntity, ParameterViolation,
                        ProfileSuspended);
```

Входной параметр **port** определяет интерфейс ATM на сетевом ресурсе. Параметр **vPI** задает значение индекса. Параметр **trafficDescriptorProfileName** определяет профиль дескриптора трафика ATM, связанного с CTP. Параметр **overbookingFactor** задает в процентах величину превышения, которая может использоваться в алгоритме управления приемом вызовов для всех PVC, имеющих одинаковое значение VPI интерфейсного порта ATM. Входной параметр **userLabel** содержит задаваемое оператором имя VPNetworkCTP. Параметр **segmentEndpoint** определяет, является ли CTP конечной точкой сегмента.

---

[3] Операция удаления приведена в 9.2.1.18.

Возвращаемое значение имеет тип **ManagedEntityIdType** и выдает уникальный идентификатор создаваемого CTP.

9.2.1.18 deleteVPNetworkCTP

Данная операция удаляет из системы управления поставщика VP к сетевой CTP. В результате ее выполнения также удаляются все управляемые объекты, автоматически созданные соответствующей операцией buildVPNetworkCTP.

Сигнатура операции **deleteVPNetworkCTP** приведена ниже:

```
void deleteVPNetworkCTP (
                    in ManagedEntityIdType vPNetworkCTP)
                    raises (UnknownManagedEntity, AccessDenied);
```

Входной параметр **vPNetworkCTP** определяет удаляемый мост.

Возвращаемое значение имеет тип **void**.

9.2.1.19 createdNodesGet

Данная операция возвращает список сетевых элементов, который был создан в процессе инициализации интерфейса.

Сигнатура операции **createdNodesGet** приведена ниже:

```
ManagedEntityIdSeqType createdNodesGet () raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **ManagedEntityIdSeqType** и содержит список узлов, созданных в результате инициализации интерфейса, описанного в документе Q834: интерфейс создания объектов.

9.2.1.20 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, когда система не предоставляет доступ к объекту интерфейса.

Исключительная ситуация **DuplicateSerialNumber** возникает, если существует другое оборудование того же типа с данным серийным номером.

Исключительная ситуация **DuplicateUserLabel** возникает, если метка пользователя, заданная в запросе, использовалась для именования другого NE или подключаемого модуля. Другими словами, система управления поставщика ответственна за политику присвоения меток пользователя, связанных с NE и подключаемыми модулями в пределах своей юрисдикции.

Исключительная ситуация **InterfaceSpeedNotChangeable** возникает, если физический порт не поддерживает новую скорость интерфейса или если скорость не может конфигурироваться.

Исключительная ситуация **InvalidEquipmentCode** возникает, если код оборудования не соответствует синтаксису, принятому оператором и поставщиком.

Исключительная ситуация **InvalidExternalTime** возникает, если задано недопустимое значение внешнего времени.

Исключительная ситуация **InvalidProtectionScheme** возникает, если сетевой ресурс не поддерживает параметры защиты, указанные в контексте перечисления портов, или если модули защиты представляют собой порты с другими характеристиками физического пути.

Исключительная ситуация **InvalidSerialNumSyntax** возникает, если синтаксис серийного номера не соответствует синтаксису поставщика.

Исключительная ситуация **InvalidSlotAssignmentList** возникает, если соответствующий блок размещения оборудования equipmentHolder не поддерживает требуемый тип подключаемого модуля.

Исключительная ситуация **InvalidUserLabelSyntax** возникает, если задаваемая метка пользователя нарушает бизнес-правила, определенные оператором и реализованные в системе управления поставщика.

Исключительная ситуация **ParameterViolation** возникает, когда VPI дублируется или не входит в допустимый диапазон.

Исключительная ситуация **ProfileSuspended** возникает, когда использование профилей, заданных при инициализации, приостановлено в системе управления поставщика оператором или OMS.

Исключительная ситуация **RemainingContainedManagedEntities** возникает, если в удаляемом объекте остались какие-либо внутренние объекты управления.

Исключительная ситуация **RemainingReservations** возникает, если остались зарезервированные ресурсы, относящиеся к удаляемому объекту.

Исключительная ситуация **RemainingSubnetworkConnections** возникает, если удаляемый объект управления имеет одно или более действующих соединений с подсетью.

Исключительная ситуация **SlotAlreadyAssigned** возникает, если запрещенный слот уже выделен.

Исключительная ситуация **UnknownManagedEntity** возникает, если указанный управляемый объект неизвестен системе управления поставщика.

Исключительная ситуация **UnknownNE** возникает, если указанный NE неизвестен системе управления поставщика.

Исключительная ситуация **UnknownProfiles** возникает, если заданное название профиля неизвестно системе управления поставщика и не может быть получено из хранилища профилей объектов.

Исключительная ситуация **UnknownSlot** возникает, если запрашиваемый слот неизвестен NE.

Исключительная ситуация **UnknownSystemTimingSource** возникает, если источник внешнего времени неизвестен системе управления поставщика или NE.

Исключительная ситуация **UnrecognizedVersion** возникает, если указанная версия оборудования не соответствует известным значениям.

9.3 Модуль Q834Common

Данный модуль содержит определения типов данных, используемых более чем одним модулем IDL в данной Рекомендации. Он также обеспечивает три группы констант. Большинство типов данных, упоминаемых в Q834Common, импортируются из Рекомендации МСЭ-Т X.780. Большинство важных аспектов этих файлов IDL содержится в определении ManagedEntityIdType. Определение извлечено из файла IDL и приведено ниже в целях обсуждения.

```
struct NamingComponentType {
     string type; // managed entity type
     string id;
};

typedef sequence<NamingComponentType> RDNType;
typedef sequence<RDNType> RDNSeqType;
typedef sequence<NameType> NameSeqType;

enum IdType {
     none,
     x780_fineGrained,
     x780_coarseGrained
};

typedef RDNType MEIdType;
```

```
struct ManagedEntityIdType {
     IdType id;
     MEIdType mEId;
};
```

ManagedEntityIdType – это возвращаемое значение для многих операций в пределах данной Рекомендации. Исследование определения этого типа данных (чтение снизу вверх) показывает, что система управления поставщика возвращает значение, содержащее ссылку либо на управляемый объект низкого или высокого уровня, либо идентификатор RDN для структуры данных в системе управления поставщика. Во всех трех случаях основной синтаксис mEId представляет собой последовательность пар именованных компонентов и соответствует синтаксису, используемому в Рекомендации МСЭ-Т X.780 для управляемых объектов, и синтаксису, используемому в Рекомендации МСЭ-Т X.780.1.

9.3.1 Модуль ProbableCauseConst и интерфейс ProbableCause

Определение модуля соответствует структуре, определенной в Рекомендации МСЭ-Т X.780, и использует тот же самый синтаксис для значений возможных причин. Она определяет несколько зависящих от технологии значений в дополнение к значениям, определенным в Рекомендации МСЭ-Т X.780.

9.3.2 Интерфейс MonitoringParameter

Данный интерфейс обеспечивает имена для отслеживаемых процессов и параметров трафика, используемых интерфейсами AlarmEventSupplier, ImpairmentPersistence и ProfileConsumer. Значения MonitoringParameter представляют собой строку значений и являются частью фильтруемых данных в структурируемом событии.

9.3.3 Интерфейс RecordSetType

Данный интерфейс обеспечивает значения данных, используемые для значений типа recordset (набор записей) интерфейсами ReportController и RecordSetMgr. Значения типа recordSet представляют собой короткие значения без знака. Значения 1–99 зарезервированы для типа данных HistoryDataType.

9.3.4 Интерфейс PhysicalLayerLoopback

Данный интерфейс обеспечивает значения данных, используемые для данных типа loopbackTestType интерфейсом TestActionPerformer. Данные типа loopbackTestType представляют собой короткие значения без знака.

9.4 Модуль ControlArchive

Система управления поставщика обеспечивает функциональность для управления регистрацией определенных групп событий, включая очистку содержимого регистрационных файлов. Привилегированный пользователь может создавать, инициализировать, приостанавливать, возобновлять и удалять файлы регистрации событий. Система управления поставщика также поддерживает функциональность для управления кратким архивированием мониторинга работы и отчетами мониторинга трафика, включая очистку содержимого этих наборов записей. Эта функция также включает в себя сообщение о статусе текущих регистрационных массивов или наборов записей статистики. Большинство кратких архивов в системе управления поставщика создаются автоматически при первоначальной установке системы. Названия этих архивов, определяемые с помощью синтаксиса ManagedEntityIdType, могут быть переданы от поставщика оператору или получены в результате вызова операции recordSetListGet со значением параметра creationModeType "initialList" и периодических вызовов операции getStatusAttributes для каждого набора записей, идентифицированного первой операцией.

9.4.1 Интерфейс RecordSetMgr

9.4.1.1 createLog

Операция CreateLog используется для создания в системе управления поставщика набора записей (RecordSet) в целях архивирования информации о событиях.

Сигнатура операции **createLog** следующая:

```
ManagedEntityIdType createLog (
              in UserLabelType recordSetUserLabel,
              in AdministrativeStateType administrativeState,
              in NameType filterName,
              in FullActionType fullAction,
              in MaxSizeType maxSize,
              in SizeThresholdType sizeThreshold)
              raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

Входной параметр **recordSetUserLabel** уникальным образом идентифицирует recordSet в системе. Входной параметр **administrativeState** определяет, инициализируется ли новый регистрационный файл для записи информации о событиях. Входной параметр **filterName** устанавливает критерии входа, определяющие, какого рода уведомления о событиях записываются в создаваемый файл. Если система управления поставщика не может распознать **filterName**, она ищет IOR для filterObject, используя **filterName** при поиске в справочнике наименований службы присваивания имен. Используя IOR, система может затем извлечь детали критерия входа из интерфейса CosNotifyFilter, доступного для службы присваивания имен. Входной параметр **fullAction** определяет поведение набора записей при достижении им максимального размера. Входной параметр **maxSize** определяет максимальный размер recordSet. Входной параметр **sizeThreshold** определяет пороговый размер RecordSet, достижение которого заставляет систему управления поставщика генерировать сигнал тревоги или соответствующее событие.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор регистрационного архива, созданного этой операцией

9.4.1.2 createArchive

Операция CreateArchive используется для вызова функции создания recordSet для хранения данных истории. Архивирование прекращается автоматически, как только архив достигает максимального размера.

Сигнатура операции **createArchive** следующая:

```
ManagedEntityIdType createArchive (
             in UserLabelType recordSetUserLabel,
             in AdministrativeStateType administrativeState,
             in RecordKindType recordKind,
             in MaxSizeType maxSize)
             raises (RecordSetExists, DuplicateUserLabel, AccessDenied);
```

Входной параметр **recordSetUserLabel** уникальным образом идентифицирует набор записей в системе управления поставщика. Входной параметр **administrativeState** определяет, должен ли создаваться новый архив для внесения соответствующих записей. Входной параметр **recordKind** определяет тип записей, которые будут храниться в наборе. Входной параметр **maxSize** устанавливает максимальный размер recordSet.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор архива, создаваемого этой операцией.

9.4.1.3 getStatusAttributes

В любой момент оператор или OMS могут посмотреть текущее состояние краткосрочного архива.

Сигнатура операции **getStatusAttributes** следующая:

```
RecordSetStatusType getStatusAttributes (
                  in ManagedEntityIdType recordSetId)
                  raises (AccessDenied, UnknownRecordSet);
```

Входной параметр **recordSetId** уникальным образом идентифицирует recordSet в системе управления поставщика.

Возвращаемое значение имеет тип **RecordSetStatusType**, который содержит текущее состояние набора записей.

9.4.1.4 suspendArchive

Когда краткосрочный архив создан и инициализирован для использования, OMS может приостановить его использование.

Сигнатура операции **suspendArchive** следующая:

```
void suspendArchive (
                    in ManagedEntityIdType recordSetId)
                    raises (AccessDenied, UnknownRecordSet);
```

Входной параметр **recordSetId** уникальным образом идентифицирует recordSet в системе управления поставщика.

Возвращаемое значение имеет тип **void.**

9.4.1.5 resumeArchive

Данная операция возобновляет запись в архив или инициализирует запись в recordSet, который был создан в заблокированном состоянии.

Сигнатура операции **resumeArchive** следующая:

```
void resumeArchive (
                    in ManagedEntityIdType recordSetId)
                    raises (UnknownRecordSet, AccessDenied);
```

Входной параметр **recordSetId** уникальным образом идентифицирует recordSet в системе управления поставщика.

Возвращаемое значение имеет тип **void**.

9.4.1.6 deleteArchive

Данная операция удаляет архив из системы управления поставщика.

Сигнатура операции **deleteArchive** следующая:

```
void deleteArchive (
             in ManagedEntityIdType recordSetId )
             raises (UnknownRecordSet, AccessDenied );
```

Входной параметр **recordSetId** определяет имя, которое привилегированный пользователь хочет использовать для идентификации архива, который будет создан при последующих вызовах.

Возвращаемое значение имеет тип **void**.

9.4.1.7 purgeArchive

Эта операция удаляет информацию, содержащуюся в указанном архиве. Но запись в архив продолжается.

Сигнатура операции **purgeArchive** следующая:

```
void purgeArchive (
               in ManagedEntityIdType recordSetId )
               raises (UnknownRecordSet, AccessDenied );
```

Входной параметр **recordSetId** определяет название, которое привилегированный пользователь хочет использовать для идентификации архива, который будет создан при последующих обращениях.

Возвращаемое значение имеет тип **void**.

9.4.1.8 selectRecords

После создания архива OMS может выбрать из него некоторые записи.

Сигнатура операции **selectRecords** следующая:

```
RecordSeqType selectRecords (
                  in FilterType selectionFilter,
                  in ManagedEntityIdType recordSetId)
                  raises (UnknownRecordSet, Timeout, NoSuchRecords,
                  AccessDenied, TooManyRecords);
```

Входной параметр **recordSetId** идентифицирует имя, которое привилегированный пользователь хочет использовать для идентификации архива, который будет создан при последующих обращениях. Входной параметр **selectionFilter** определяет соответствующий набор записей в соответствии со структурой данных.

Возвращаемое значение имеет тип **RecordSeqType** и содержит запрошенную информацию.

9.4.1.9 recordSetListGet

Данная операция позволяет OMS получить полный листинг наборов записей, управляемых системой управления поставщика.

Сигнатура операции **recordSetListGet** следующая:

```
ManagedEntityIdSeqType recordSetListGet (
                       in CreationModeType creationMode)
                       raises (AccessDenied);
```

Входной параметр **creationMode** определяет способ создания архива, то есть операторов или автоматически, в процессе инициализации систему управления поставщика.

Возвращаемое значение имеет тип **ManagedEntityIdSeqType** и перечисляет имена краткосрочных архивов системы.

9.4.1.10 changeUserLabel

После создания краткосрочного архива OMS может изменить присвоенную ему метку пользователя.

Сигнатура операции **changeUserLabel** следующая:

```
void changeUserLabel(
             in ManagedEntityIdType recordSetId,
             in UserLabelType newUserLabel)
             raises (UnknownRecordSet, AccessDenied, DuplicateUserLabel);
```

Входной параметр **recordSetId** определяет архив. Входной параметр **newUserLabel** определяет новое имя указанного архива.

Возвращаемое значение имеет тип **void**.

9.4.1.11 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, когда система не дает доступа к объекту интерфейса.

Исключительная ситуация **DuplicateUserLabel** возникает, если метка пользователя, указанная в запросе, используется для именования другого архива. Другими словами, система управления поставщика ответственна за соблюдение политики присвоения меток пользователя наборам данных в пределах своей юрисдикции.

Исключительная ситуация **LockedAlready** возникает, если уже установлено значение параметра administrativeState – "заблокировано".

Исключительная ситуация **NoSuchRecords** возникает, если в указанном наборе записей ни одна не соответствует критерию выборки.

Исключительная ситуация **RecordSetExists** возникает, если архив, определенный параметрами запроса на создание, уже существует в системе управления поставщика.

Исключительная ситуация **Timeout** возникает, если длительность процесса до завершения достигает величины определенного системой тайм-аута.

Исключительная ситуация **TooManyRecords** возникает, если количество записей, выбранных в ответ на запрос, превышает предварительно определенный раздел[4].

Исключительная ситуация **UnknownRecordSet** возникает, если указанный набор записей неизвестен системе управления поставщика.

9.5 Модуль SoftwareDownload

Процесс загрузки программ состоит из четырех фаз: доставки, распределения, установки (подтверждения) и активации. Система управления поставщика с помощью данного модуля поддерживает для NE три стадии загрузки программ: первоначальную загрузку, обновление и поддержку изменений (патчей). Система может принять одновременно запросы, относящиеся к одному или нескольким NE. Вся работа по управлению программным обеспечением может быть индивидуально спланирована. Любой запрос на первоначальную загрузку программ сопровождается подтверждениями безопасности, в которых системе управления поставщика разрешается связываться с соответствующим сервером. Использование конкретных подтверждений безопасности зависит от конкретной реализации: Если загрузка программ резидентна на EMS и EMS будет передавать их (посредством пересылки файлов) на NE, то подтверждения устанавливаются для NE. Если программы скачиваются на EMS, а NE затем забирает их, то подтверждения устанавливаются для EMS. Если программы загружаются в отдельное хранилище, то подтверждения безопасности устанавливаются для этого отдельного хранилища, а NE затем забирает программы из него.

Запросы на управление программным обеспечением могут вызывать создание объектов отслеживания. Жизненный цикл такого объекта должен продолжаться до тех пор, пока все соответствующие действия успешно или неуспешно завершатся. Объект удаляется только после предварительно определенного периода сохранения, и длительность этого периода согласуется вне рамок данного интерфейса. Период сохранения объекта отслеживания должен быть достаточен для включения всех необходимых действий. Если объект отслеживания автоматически удаляется системой управления поставщика, то возврат в исходное состояние рассматривается как часть процесса предыдущей загрузки программ. OMS также может удалить, в случае необходимости, объект отслеживания до его автоматической ликвидации.

Система управления поставщика создает для каждого действия по загрузке программ, включая доставку, подтверждение и активацию, регистрационную запись независимо от результата. Эта запись содержит детальный результат всех контролируемых событий, включая время запуска/остановки и индикацию успешного или неуспешного завершения.

Система управления поставщика поддерживает текущий листинг всех выполняемых действий по загрузке программ.

Данный модуль также включает в себя поддержку получения с установленных NE информации о версиях оборудования и программного обеспечения. Также можно проверить, что пакет программного обеспечения может быть загружен на NE или на указанный подключаемый модуль.

9.5.1 Интерфейс DownloadMgr

9.5.1.1 deliverDistSWGlobal

Данная операция запрашивает систему управления поставщика о первичной загрузке программ с исходной машины в целях обновления программного обеспечения или поддержки изменений (патчей) на NE. Система управления поставщика может принять запросы для одного или нескольких NE одновременно. Данная операция означает, что система пытается выполнить доставку и распределение программ для всех указанных элементов. Она настроена так, что пытается доставить и распределить программы всем указанным элементам, даже если на одном из них процесс завершился неудачно.

---

[4] Этот размер должен быть согласован в дальнейшем между поставщиком и оператором.

Для фиксации как успешных, так и неуспешных попыток доставки и распределения используется регистрационный файл. Операция SoftwareDownloadTrackingObject остается доступной для подтверждения и попыток активации только на узлах, на которых доставка и распределение завершились успешно. Операция SoftwareDownloadTrackingObject сохраняется и после стадии успешной активации (в частности, для возврата в исходное состояние, если потребуется).

Следовательно, если появился новый ONT, или если нарушенное волоконно-оптическое соединение восстановлено, или если установлен новый подключаемый модуль и в каждом из этих случаев активная загрузка программ отличается от описанной для данной операции, то система управления поставщика (или сетевой ресурс) будут ответственны за автоматическое обновление программ.

Сигнатура операции **deliverDistSWGlobal** приведена ниже:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in ManagedEntityIdSeqType deliverDistTargets)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
                        SourceUnreachable, UnknownSoftwareLoad, Timeout,
                        AccessDenied, DeniedAccess);
```

Входной параметр **softwareSet** определяет загружаемое программное обеспечение и где оно расположено (имена файлов с полным маршрутом). Входной параметр **softwareSourceAddr** задает адрес DCN сервера, где расположен набор загружаемых программ. Входные параметры **userId** и **password** обеспечивают механизм входа на сервер-источник (softwareSource) (предполагается, что нужны соответствующие подтверждения безопасности). Входной параметр **deliverDistTargets** указывает список OLT, куда должны быть доставлены программы.

Возвращаемое значение имеет тип **SoftwareDownloadTrackingObjectIdType** и содержит ссылку, которая будет использоваться при подтверждении, проверке состояния процесса доставки и распределения. Результат загрузки программ фиксируется в системе управления поставщика.

9.5.1.2 deliverDistSWSpecific

Данная операция представляет собой то же самое, что и deliverDistSWGlobal, за исключением того, что относится к одному NE/подключаемому модулю/слоту, как описано во входном параметре distributionTarget.

Сигнатура операции **deliverDistSWSpecific** приведена ниже:

```
SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
                        in FilenameSeqType softwareSet,
                        in DCNAddressType softwareSourceAddr,
                        in UserIdType userId,
                        in PasswordType password,
                        in TargetType deliverDistTarget)
                        raises (CommFailure, UnrecognisedTarget,
                        InsufficientMemory, SoftwareLoadHWMismatch,
                        SourceUnreachable, UnknownSoftwareLoad, Timeout,
                        AccessDenied, DeniedAccess);
```

Входной параметр **softwareSet** определяет программное обеспечение и его местонахождение (имена файлов с полным маршрутом). Входной параметр **softwareSourceAddr** задает адрес DCN сервера, где расположен загружаемый набор программ. Входные параметры **userId** и **password** обеспечивают механизм входа на softwareSource (предполагается, что нужны соответствующие подтверждения безопасности). Входной параметр **deliverDistTarget** указывает адресата установки на уровне NE/PlugInUnit/слот. Компоненты значения этого параметра приведены в таблице 3.

Таблица 3/Q.834.4 – Детали параметра deliverDistTarget

Имя поля	Описание
containingSystem	Система идентифицируется посредством идентификатора управляемого объекта конечного OLT.
containingNE	Идентифицирует конечный сетевой ресурс. Если значение представляет собой пустую последовательность, то программы распределяются всем NE в системе.
plugInUnitType	Определяет тип подключаемого модуля (plugInUnitType). Если это значение представляет собой пустую строку, то распределение происходит всем типам plugInUnitTypes на containingNE.
slot	Определяет слот. Если это значение представляет собой пустую последовательность, то используется любой подходящий слот.

Возвращаемое значение имеет тип **SoftwareDownloadTrackingObjectIdTypeId** и содержит коррелированную ключевую ссылку, которая будет использоваться при подтверждении, активации или проверке состояния процесса доставки и распределения. Результат загрузки программ регистрируется системой управления поставщика.

9.5.1.3 deleteSoftwareDownloadTrackingObject

Данная операция позволяет OMS уведомлять систему управления поставщика о том, что указанный объект установки программ больше не нужен.

Сигнатура операции **deleteSoftwareDownloadTrackingObject** приведена ниже:

```
void deleteSoftwareDownloadTrackingObject (
            in SoftwareDownloadTrackingObjectIdType   id)
            raises (UnknownSoftwareDownloadTrackingObject, AccessDenied,
            SoftwareTrackingObjectInUse);
```

Входной параметр **id** определяет объект загрузки программ.

Возвращаемое значение имеет тип **void**.

9.5.1.4 commit

Данная операция запрашивает у системы управления поставщика установку (подтверждение) загруженных программ в месте назначения.

Сигнатура операции **commit** приведена ниже:

```
void commit (
      in SoftwareDownloadTrackingObjectIdType id,
      in TargetType commitTarget)
      raises (InstallationFailure, UnknownSoftwareDownloadTrackingObject,
      AccessDenied, UnrecognisedTarget);
```

Входной параметр **id** содержит ссылку на конечный объект установки загруженных программ. Входной параметр **commitTarget** указывает конечный объект на уровне NE/PlugInUnit/слот. Параметр commitTarget может быть подмножеством начального списка объектов назначения.

Возвращаемое значение имеет тип **void**.

9.5.1.5 activate

Данная операция активирует установленные программы на конечном объекте назначения.

Сигнатура операции **activate** приведена ниже:

```
void activate (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType activateTarget)
```

```
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget);
```

Входной параметр **id** содержит ссылку на конечный объект установки загруженных программ. Входной параметр **activateTarget** указывает конечный объект на уровне NE/PlugInUnit/слот. Параметр activateTarget может быть подмножеством начального списка объектов назначения.

Возвращаемое значение имеет тип **void**.

9.5.1.6 revert

Данная операция активирует старое программное обеспечение на конечном объекте назначения. Операцию revert имеет смысл вызывать только после активации новых загруженных программ. Параметр SoftwareDownloadTrackingObjectId указывает на самую новую версию ранее загруженных программ на заданном объекте назначения. Если операция revert завершается успешно, то активная версия становится запасной версией.

Сигнатура операции **revert** приведена ниже:

```
void revert (
            in SoftwareDownloadTrackingObjectIdType id,
            in TargetType revertTarget)
            raises (UnknownSoftwareDownloadTrackingObject,
            SoftwareNotYetInstalled, ActivationFailure, AccessDenied,
            UnrecognisedTarget, InvalidSoftwareTrackingObject);
```

Входной параметр **id** задает ссылку на конечный объект установки программного обеспечения. Входной параметр **revertTarget** определяет указанный объект назначения на уровне NE/PlugInUnit/слот. Параметр revertTarget может быть подмножеством первоначального списка объектов назначения.

Возвращаемое значение имеет тип **void**.

9.5.1.7 getStatus

Данная операция запрашивает состояние процесса загрузки программ.

Сигнатура операции **getStatus** приведена ниже:

```
DownloadStatusSeqType getStatus (
              in SoftwareDownloadTrackingObjectIdType id)
              raises (UnknownSoftwareDownloadTrackingObject, AccessDenied);
```

Входной параметр **id** задает ссылку на конечный объект установки программного обеспечения.

Возвращаемое значение имеет тип **DownloadStatusSeqType** и содержит информацию о состоянии выполнения процесса загрузки программ для данного объекта назначения.

9.5.1.8 scheduleDeliverDist

Эта операция планирует доставку и распределение программ указанным объектам назначения. Распределение программ между объектами назначения зависит от конкретной реализации поставщика.

Сигнатура операции **scheduleDeliverDist** приведена ниже:

```
SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
              in FilenameSeqType softwareSet,
              in DCNAddressType softwareSourceAddr,
              in UserIdType userId,
              in PasswordType password,
              in ManagedEntityIdSeqType deliverDistTargets,
              in GeneralizedTimeType deliverDistStartTime)
              raises (SoftwareLoadHWMismatch, AccessDenied, InvalidStartTime );
```

Входной параметр **softwareSet** определяет загружаемые программы и их местонахождение (имена файлов с полным маршрутом). Входной параметр **softwareSourceAddr** задает адрес DCN сервера, где находятся загружаемые программы. Входные параметры **userId** и **password** обеспечивают механизм входа на softwareSource (предполагается, что нужны соответствующие подтверждения безопасности). Входной параметр **deliverDistTargets** указывает список OLT, куда должны быть доставлены программы. Входной параметр **deliverDistStartTime** указывает планируемое время запуска.

Возвращаемое значение имеет тип **SoftwareDownloadTrackingObjectIdType** и содержит ссылку на запланированный процесс, которая может быть использована для слежения за ходом процесса или для его отмены.

9.5.1.9 scheduleCommit

Данная операция планирует установку (подтверждение) программ на предварительно определенные объекты назначения.

Сигнатура операции **scheduleCommit** приведена ниже:

```
void scheduleCommit (
               in SoftwareDownloadTrackingObjectIdType
                deliverDistSoftwareDownloadTrackingObjectId,
               in GeneralizedTimeType commitStartTime)
               raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

Входные параметры **deliverDistSoftwareDownloadTrackingObjectId** задают ссылку на контролируемый объект, куда будут загружаться программы. Входной параметр **commitStartTime** задает планируемое время запуска указанных действий.

Возвращаемое значение имеет тип **void**.

9.5.1.10 scheduleActivate

Данная операция планирует активацию программ на предварительно определенных объектах.

Сигнатура операции **scheduleActivate** приведена ниже:

```
void scheduleActivate (
               in SoftwareDownloadTrackingObjectIdType id,
               in GeneralizedTimeType activateStartTime)
               raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled, AccessDenied, InvalidStartTime );
```

Входной параметр **id** задает ссылку на объект назначения, куда доставляются программы. Входной параметр **activateStartTime** задает время запуска указанного действия.

Возвращаемое значение имеет тип **void**.

9.5.1.11 cancelScheduledSoftwareActivity

Данная операция отменяет все последующие действия по загрузке программ, относящихся к данному контролируемому объекту.

Сигнатура операции **cancelScheduledSoftwareActivity** приведена ниже:

```
void cancelScheduledSoftwareActivity (
                    in SoftwareDownloadTrackingObjectIdType id)
                    raises (UnknownSoftwareDownloadTrackingObject,
                    ActivityCompleted, ActivityInProgress, AccessDenied);
```

Входные параметры **id** задают ссылку на контролируемый объект, куда загружаются программы.

Возвращаемое значение имеет тип **void**.

9.5.1.12 scheduledSoftwareDownloadTrackingObjectListGet

Данная операция возвращает список незавершенных запланированных действий по загрузке программ.

Сигнатура операции **scheduledSoftwareDownloadTrackingObjectListGet** приведена ниже:

```
SoftwareDownloadTrackingObjectIdSeqType
scheduledSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **SoftwareDownloadTrackingObjectIdSeqType** и содержит список незавершенных запланированных действий.

9.5.1.13 onDemandSoftwareDownloadTrackingObjectListGet

Данная операция возвращает список незавершенных незапланированных действий по загрузке программ.

Сигнатура операции **onDemandSoftwareDownloadTrackingObjectListGet** приведена ниже:

```
SoftwareDownloadTrackingObjectIdSeqType
onDemandSoftwareDownloadTrackingObjectListGet () raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **SoftwareDownloadTrackingObjectIdSeqType** и содержит список незавершенных незапланированных действий.

9.5.1.14 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, когда система не дает доступа к объекту интерфейса.

Исключительная ситуация **ActivityCompleted** возникает, когда действие по загрузке программ выполняется и не может быть отменено.

Исключительная ситуация **ActivationFailure** возникает, если процесс активации программного обеспечения завершился неудачно, даже если сами программы установлены (т. е. подтверждение выполнено успешно).

Исключительная ситуация **ActivityInProgress** возникает, если действие по загрузке программ инициировано и не может быть отменено.

Исключительная ситуация **CommFailure** возникает, если DCN между системой управления поставщика и OLT или связь между OLT и исходным ONT не работает.

Исключительная ситуация **DeniedAccess** возникает, если доступ к NE запрещен из-за ограничений управления доступом.

Исключительная ситуация **InstallationFailure** возникает, если процесс установки программ завершился неудачно.

Исключительная ситуация **InsufficientMemory** возникает, если для загрузки программ на NE недостаточно памяти.

Исключительная ситуация **InvalidSoftwareTrackingObject** возникает, если отслеживаемый объект назначения, на который производится установка программ, не связан с последней версией программ, загруженных на NE.

Исключительная ситуация **InvalidStartTime** возникает, если время запуска меньше текущего системного времени.

Исключительная ситуация **SoftwareLoadHWMismatch** возникает, если требуемое программное обеспечение не может быть загружено по причине того, что оборудование не поддерживает загрузку данных программ.

Исключительная ситуация **SoftwareNotYetInstalled** возникает, когда запрошена активация, а программы еще не установлены.

Исключительная ситуация **SoftwareTrackingObjectInUse** возникает, когда есть незавершенные действия по загрузке программ для данного объекта и он не может быть удален.

Исключительная ситуация **SourceUnreachable** возникает, если сервер, на котором хранятся программы, предназначенные для загрузки, не доступен для OLT.

Исключительная ситуация **Timeout** возникает, если длительность процесса достигает определенного системой тайм-аута, а процесс еще не завершен.

Исключительная ситуация **UnknownSoftwareLoad** возникает, когда не удается найти указанный набор программ.

Исключительная ситуация **UnknownSoftwareDownloadTrackingObject** возникает, когда процесс загрузки программ неизвестен системе управления поставщика.

Исключительная ситуация **UnrecognizedTarget** возникает, когда требуемое программное обеспечение на безопасном файловом сервере неизвестно системе управления поставщика.

9.5.2 Интерфейс VersionRepository

9.5.2.1 retrieveVersions

Данная операция возвращает всю информацию о версии (программного обеспечения и оборудования) сетевого ресурса.

Сигнатура операции **retrieveVersions** приведена ниже:

```
VersionsSeqType retrieveVersions (
                  in ManagedEntityIdType containingManagedEntityId)
                  raises (CommFailure, UnknownManagedEntity, AccessDenied);
```

Входной параметр **containingManagedEntityId** определяет сетевой ресурс.

Возвращаемое значение имеет тип **VersionsSeqType** и содержит листинг оборудования и версий программного обеспечения, относящихся к данному сетевому ресурсу.

9.5.2.2 validateNEVersion

Данная операция запрашивает у системы управления поставщика подтверждение, что предлагаемое программное обеспечение совместимо с NE.

Сигнатура операции **validateNEVersion** приведена ниже:

```
boolean validateNEVersion (
            in ManagedEntityIdType managedEntityId,
            in VersionType proposedSoftware)
            raises (UnknownNE, AccessDenied);
```

Входные параметры **managedEntityId** идентифицируют NE.

Входной параметр **proposedSoftware** идентифицирует предлагаемое программное обеспечение, которое должно быть подтверждено.

Возвращаемое значение имеет тип **boolean**.

9.5.2.3 validatePlugInVersion

Данная операция запрашивает у системы управления поставщика подтверждение, что предлагаемое программное обеспечение совместимо с подключаемым модулем (plugInUnit).

Сигнатура операции **validatePlugInVersion** приведена ниже:

```
boolean validatePlugInVersion (
              in ManagedEntityIdType plugInUnitId,
              in VersionType proposedSoftware)
              raises (UnknownManagedEntity, AccessDenied);
```

Входной параметр **plugInUnitId** идентифицирует plugInUnit.

Входной параметр **proposedSoftware** идентифицирует предлагаемое программное обеспечение, которое должно быть подтверждено.

Возвращаемое значение имеет тип **boolean**.

9.5.2.4 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, когда система не дает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, когда не работает DCN между системой управления поставщика и OLT или связь между OLT и исходным ONT.

Исключительная ситуация **UnknownManagedEntity** возникает, если идентификатор plugInUnit или порт неизвестен системе управления поставщика.

9.6 Модуль EventPublisher

При получении информации об обработанной конфигурации, функционировании или отказах, формируемой в разных ситуациях в системе управления поставщика и основанной на правилах, относящихся к публикации, система ставит ее в очередь и направляет в каналы всем заинтересованным пользователям, включая операторов и OMS.

9.6.1 Интерфейс AlarmEventSupplier

Целью интерфейса является оповещение о событиях тревоги операторов системы управления через службу уведомлений OMG. Этот интерфейс не имеет операций. Однако он обеспечивает отображение фиксированного заголовка, а также фильтруемых данных для структурируемых объектов событий, которые используются для доставки информации о событиях по каналу событий службы уведомлений OMG. Оба набора отображений соответствуют принципам, изложенным в Рекомендации МСЭ-Т X.780 для интерфейса уведомлений.

В фиксированном заголовке **domain_type** установлен в значение "telecommunications", **type_name** установлен в значение "Alarm", а **event_name** установлен в постоянную строку, имеющую одно из следующих значений: "Communications Alarm", "Environmental Alarm", "Equipment Alarm", "Processing Error Alarm" или "Quality of Service Alarm".

Отображение в фильтруемые данные состоит из пар элементов. Первый компонент пары – строковый идентификатор имени данных, а второй – значение элемента данных. Строковые идентификаторы являются константами, определенными в данном интерфейсе. Кроме того, фильтруемые пары данных должны появляться в определенном порядке.

Порядок фильтруемых элементов следующий:

- AlarmEmittingMEId
- EventTime
- ProbableCause
- SpecificProblems
- PerceivedSeverity
- ServiceAffectingInd
- BackUpStatus
- BackUpManagedEntityId
- ThresholdInfo
- NotificationIdentifier
- CorrelatedNotifications
- StateChangeDefinition
- AdditionalText

Значение ProbableCause имеет синтаксис типа ProbableCauseType и принимает одно из конкретных значений, определенных в Рекомендации МСЭ-Т X.780 или q834_4::Q834Common::ProbableCauseConst.

Значение ThresholdInfo имеет синтаксис типа MonitoredParameterType и принимает одно из конкретных значений, определенных в q834_4::Q834Common::MonitoredParameter. Это значение может быть пустой строкой для всех имен событий (event_names), кроме "QualityOfServiceAlarm".

Синтаксис и интерпретация всех других типов данных и сведения об их использовании приведены в Рекомендации МСЭ-Т X.780.

9.6.2 SecurityEventSupplier

Цель этого интерфейса – сообщить системе управления оператора о событиях, связанных с безопасностью, через службу уведомлений OMG. Этот интерфейс не имеет операций. Однако он обеспечивает отображение фиксированного заголовка, а также фильтруемых данных для структурируемых объектов событий, которые используются для доставки информации о событиях по каналу событий службы уведомлений OMG. Оба набора отображений соответствуют принципам, изложенным в Рекомендации МСЭ-Т X.780 для интерфейса уведомлений.

В фиксированном заголовке поле **domain_type** установлено в значение "telecommunications", поле **type_name** – в значение "SecurityEvent", а поле **event_name** является постоянной строкой, имеющей одно из следующий значений: "IntegrityViolation", "OperationalViolation", "PhysicalViolation", "SecurityEventViolation" или "TimeDomainViolation".

Отображение фильтруемых данных состоит из пар элементов. Первый компонент пары является строковым идентификатором имени данных, а второй – значением элемента данных. Строковые идентификаторы являются константами, определенными в данном интерфейсе. Кроме того, пары фильтруемых данных должны появляться в определенном порядке.

Порядок фильтруемых элементов следующий:

- EventEmittingMEId
- EventTime
- SecurityAlarmCause
- SecurityAlarmDetector
- ServiceUser
- ServiceProvider
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

Синтаксис типов данных и сведения об их использовании приведены в Рекомендации МСЭ-Т X.780.

9.6.3 DiscoveryEventSupplier

Цель этого интерфейса – сообщить системе управления оператора об изменениях в установленном оборудовании через службу уведомлений OMG. Данный интерфейс не имеет операций. Однако он обеспечивает отображение фиксированного заголовка, а также фильтруемых данных для структурируемых объектов событий, которые используются для доставки информации о событиях по каналу событий службы уведомлений OMG.

В фиксированном заголовке поле **domain_type** установлено в "telecommunications", поле **type_name** – в "DiscoveryEvent", а поле **event_name** представляет собой постоянную строку, имеющую одно из следующих значений: "ManagedEntityCreation" или "ManagedEntityDeletion".

Отображение фильтруемых данных состоит из пар элементов. Первый компонент пары – строковый идентификатор имени данных, в второй – значение элемента данных. Строковые идентификаторы являются константами, определенными в данном интерфейсе. Кроме того, пары фильтруемых данных должны появляться в определенном порядке.

Порядок фильтруемых элементов для значения event_name "ManagedEntityCreation" следующий:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

Значение ManagedEntityType имеет синтаксис типа EquipmentType, который определяет тип раскрываемых данных описи. Интерфейс определяет константы для разных типов NE, а также plugInUnits, equipmentHolders и программного обеспечения.

Значение EventTime имеет синтаксис типа GeneralizedTimeType и ссылается на момент, когда в сети было определено условие раскрытия.

Значение ManagedEntityAttributeValues имеет синтаксис MEstruct. Однако тип данных для этого значения является одной из набора структур, определенных в модуле. Элемент данных ManagedEntityType определяет тип структуры, передаваемой в этом значении в составе объекта с информацией о событии.

Значение NotificationIdentifier имеет синтаксис типа NotificationIdentifierType и содержит последовательный номер события. Значение CorrelatedNotifications имеет синтаксис типа CorrelatedNotificationType и содержит список номеров для других уведомлений о событиях, формируемых системой управления поставщика в соответствующих условиях, связанных с описью оборудования. Если нет соответствующих уведомлений, то выдается пустой набор значений.

Наконец, значение AdditionalText имеет синтаксис в форме строки. Этот элемент данных представляет собой место для передачи любой дополнительной текстовой информации от системы управления поставщика, относящейся к условиям изменения описи оборудования. Если дополнительной информации нет, то передается пустая строка.

Порядок фильтруемых элементов для значения event_name "ManagedEntityDeletion" следующий:

- ManagedEntityType
- EventTime
- ManagedEntityAttributeValues
- NotificationIdentifier
- CorrelatedNotifications
- AdditionalText

Значение ManagedEntityType имеет синтаксис типа EquipmentType, который указывает тип удаляемых данных описи. Интерфейс определяет константы для различных типов NE, а также для plugInUnits и equipmentHolders.

Значение EventTime имеет синтаксис типа GeneralizedTimeType и ссылается на момент, когда в сети определено условие удаления.

Значение ManagedEntityAttributeValues имеет синтаксис MEstruct. Однако тип данных для этого значения является одной из структур, определенный в модуле. Элемент данных ManagedEntityType определяет тип структуры, которая передается в этом значении в составе структурированного объекта с информацией о событии.

Значение NotificationIdentifier имеет синтаксис типа NotificationIdentifierType и содержит последовательный номер события. Значение CorrelatedNotifications имеет синтаксис типа CorrelatedNotificationType и содержит список номеров для уведомлений о других событиях, формируемых системой управления поставщика и при возникновении соответствующих условий изменения описи оборудования. Если уведомлений нет, то выдается пустой набор значений.

Наконец, значение AdditionalText имеет синтаксис в форме строки. Этот элемент данных представляет собой место для передачи любой дополнительной текстовой информации от системы управления поставщика, относящейся к условиям изменения описи оборудования. Если дополнительной информации нет, то передается пустая строка.

Установка или удаление подключаемого модуля всегда создает уведомление независимо от состояния окружения этого модуля.

9.7 Модуль MIBTransfer

Данный модуль управляет процессом сбора данных о конфигурации системы, которые будут использоваться при ее восстановлении в случае катастрофического отказа. Сбор данных о конфигурации может производиться как в реальном времени, так и на основе расписания. Модуль также обеспечивает информацию о состоянии выполняющихся процессов резервного копирования или восстановления. Результаты этих процессов фиксируются системой управления поставщика. Каждый запрос на резервное копирование или восстановление данных конфигурации сопровождается проверкой условий безопасности на предмет того, разрешена ли системе управления поставщика или OLT связь с внешним сервером. Когда выполняется резервное копирование, все запросы, относящиеся к соответствующей системе OLT, сбрасываются или блокируются. Объекты слежения за передачей автоматически удаляются системой, как только передача соответствующего файла (файлов) завершена и результат (успешный или неуспешный) зафиксирован в регистрационном файле.

9.7.1 Интерфейс MIBMover

9.7.1.1 startBackup

Данная операция инициирует немедленное резервное копирование данных конфигурации из системы и /или системы управления поставщика на внешний сервер.

Сигнатура операции **startBackup** приведена ниже:

```
TransferTrackingObjectIdType startBackup (
                    in ManagedEntityIdType nEManagedEntityId,
                    in DCNAddressType destinationServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType destinationFile,
                    in boolean overwriteExistingFile)
                    raises (UnknownNE, UnknownDestinationServer, CommFailure,
                    EquipmentFailure, DeniedAccess, AccessDenied);
```

Входной параметр **nEManagedEntityId** определяет систему, для которой выполняется резервное копирование. Входной параметр **destinationServerAddr** определяет данные для связи с сервером, на который производится резервное копирование. Входной параметр **userId** определяет вход пользователя на сервер резервного копирования. Входной параметр **password** определяет пароль доступа к серверу. Входной параметр **destinationFile** определяет полное расположение каталога для файла резервного копирования. Наконец, параметр **overwriteExistingFile** показывает, должна ли процедура копирования позволять перезаписывать ранее созданный файл резервной копии в том же самом каталоге.

Возвращаемое значение имеет тип **TransferTrackingObjectIdType** и выдает ключ корреляции для отслеживания состояния выполнения процесса копирования.

9.7.1.2 getBackupStatus

Данная операция позволяет получать данные о состоянии процесса копирования.

Сигнатура операции **getBackupStatus** приведена ниже:

```
StatusAttributeSeqType getBackupStatus (
                    in TransferTrackingObjectIdType id)
                    raises (UnknownBackupProcess,AccessDenied);
```

Входной параметр **id** идентифицирует процесс копирования.

Возвращаемое значение имеет тип **StatusAttributeSeqType** и выдает состояние предварительно запрошенного процесса резервного копирования.

9.7.1.3 scheduleBackup

Данная операция планирует процессы резервного копирования.

Сигнатура операции **scheduleBackup** приведена ниже:

```
TransferTrackingObjectIdType scheduleBackup(
               in ManagedEntityIdType nEManagedEntityId,
               in UserIdType userId,
               in PasswordType password,
               in UserLabelType schedulerName,
               in DCNAddressType destinationServerAddr,
               in FilenameType destinationFile,
               in boolean overwriteExistingFile)
               raises (UnknownNE, UnknownScheduler, UnknownDestinationServer,
               InvalidScheduler, AccessDenied);
```

Входной параметр **nEManagedEntityId** определяет OLT, для которой будет выполняться резервное копирование. Входной параметр **schedulerName** представляет собой имя расписания, используемого для копирования. Входной параметр **userId** определяет вход пользователя на сервер. Входной параметр **password** – пароль доступа к серверу. Входной параметр **destinationServerAddr** определяет адрес DCN сервера, где будут храниться данные копирования. Входной параметр **destinationFile** определяет полное расположение каталога для файла копии. Наконец, параметр **overwriteExistingFile** показывает, должна ли процедура копирования позволять перезаписывать ранее созданный файл резервной копии в том же самом каталоге.

Возвращаемое значение имеет тип **TransferTrackingObjectIdType** и содержит ключ корреляции, который будет использоваться при попытках отслеживания процесса не в реальном времени в последующие моменты.

9.7.1.4 modifyBackupSchedule

Данная операция отменяет все последующие процессы копирования для системы на основе расписания. Она не прерывает уже выполняющийся процесс.

Сигнатура операции **modifyBackupSchedule** приведена ниже:

```
void modifyBackupSchedule (
               in TransferTrackingObjectIdType id,
               in UserLabelType newSchedulerName)
               raises (UnknownBackupProcess, AccessDenied,UnknownScheduler,
               InvalidScheduler);
```

Входной параметр **id** идентифицирует запланированный процесс, который должен быть изменен. Входной параметр **newSchedulerName** определяет новые временные триггеры.

Возвращаемое значение имеет тип **void**.

9.7.1.5 cancelScheduledBackup

Данная операция отменяет все последующие процессы копирования для системы на основе расписания. Она не прерывает уже выполняющийся процесс.

Сигнатура операции **cancelScheduledBackup** приведена ниже:

```
void cancelScheduledBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, AccessDenied);
```

Входной параметр **id** определяет запланированный процесс, который должен быть отменен.

Возвращаемое значение имеет тип **void**.

9.7.1.6 abortBackup

Данная операция прерывает процесс копирования независимо от того, выполняется он по расписанию или нет. На последующие процессы копирования эта операция не влияет.

Сигнатура операции **abortBackup** приведена ниже:

```
void abortBackup (
            in TransferTrackingObjectIdType id)
            raises (UnknownBackupProcess, CommFailure, EquipmentFailure,
            AccessDenied);
```

Входной параметр **id** определяет процесс копирования, который должен быть прерван.

Возвращаемое значение имеет тип **void**.

9.7.1.7 startRestore

Операция **startRestore** обеспечивает функциональность восстановления системы на основе сохраненной резервной копии данных конфигурации.

Сигнатура операции **startRestore** приведена ниже. Данные конфигурации расположены на внешнем сервере безопасности.

```
TransferTrackingObjectIdType startRestore (
                    in ManagedEntityIdType nEManagedEntityId,
                    in DCNAddressType sourceServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType sourceFile)
                    raises ( UnknownNE, CommFailure, EquipmentFailure,
                    UnknownSourceServer, DeniedAccess,
                    SoftwareLoadHardwareMismatch, AccessDenied );
```

Входной параметр **nEManagedEntityId** определяет OLT, которое будет восстановлено. Входной параметр **sourceServerAddr** определяет коммуникационный сетевой адрес сервера резервной копии. Входной параметр **userId** определяет вход пользователя на сервер. Входной параметр **password** – пароль доступа к серверу. Входной параметр **sourceFile** определяет файл, в котором сохранены данные и из которого будет происходить восстановление.

Возвращаемое значение имеет тип **TransferTrackingObjectIdType** и содержит ключ корреляции, используемый при попытках отслеживания состояния запрошенной процедуры восстановления.

9.7.1.8 getRestoreStatus

Данная операция выдает состояние процесса восстановления.

Сигнатура операции **getRestoreStatus** приведена ниже:

```
StatusAttributeSeqType getRestoreStatus (
                        in TransferTrackingObjectIdType id)
                        raises (UnknownRestoreProcess, AccessDenied);
```

Входной параметр **id** определяет процесс восстановления.

Возвращаемое значение имеет тип **StatusAttributeSeqType** и выдает состояние выполнения процесса восстановления.

9.7.1.9 transferTrackingObjectIdListGet

Данная операция выдает список незавершенных системных модулей преобразования (MIB transfers), как резервных копирований, так и восстановлений.

Сигнатура операции **transferTrackingObjectIdListGet** приведена ниже:

```
TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
                    raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **TransferTrackingObjectIdSeqType** и содержит список незавершенных системных модулей преобразования, как резервных копирований, так и восстановлений.

9.7.1.10 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, когда система не дает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, когда DCN между системой управления поставщика и OLT или связь между OLT и исходным ONT не работает.

Исключительная ситуация **DeniedAccess** возникает, если доступ к NE запрещен.

Исключительная ситуация **EquipmentFailure** возникает, если оборудование, с которого была сделана резервная копия, находится в состоянии отказа.

Исключительная ситуация **InvalidScheduler** возникает, если данное расписание не пригодно для выполнения операции или устарело.

Исключительная ситуация **SoftwareLoadHardwareMismatch** возникает, если программное обеспечение не соответствует оборудованию, на котором оно должно восстанавливаться.

Исключительная ситуация **UnknownBackupProcess** возникает, если не найден заданный TransferTrackingObjectId, относящийся к процессу копирования.

Исключительная ситуация **UnknownDestinationServer** возникает, если не найден IP-адрес сервера.

Исключительная ситуация **UnknownNE** возникает, если OLT неизвестно системе управления поставщика.

Исключительная ситуация **UnknownRestoreProcess** возникает, если неизвестен заданный TransferTrackingObjectId, относящийся к процессу восстановления.

Исключительная ситуация **UnknownScheduler** возникает, если не найдено заданное имя расписания.

Исключительная ситуация **UnknownSourceServer** возникает, если исходный сервер неизвестен системе управления поставщика.

9.8 Модуль PerformanceManager

Система управления поставщика должна обеспечивать активацию или деактивацию данных о текущей работе или об измерениях трафика, передаваемых на отдельные конечные пункты на NE в соответствии с требованиями оператора или OMS к установленным OLT. Требования к системе управления поставщика, содержащиеся в данной Рекомендации, включают также реализацию установок пороговых значений и описывают автоматические отчеты об измерениях параметров

функционирования для системы управления поставщика, когда пороговые значения пройдены. Когда возникает необходимость установки сигналов о пересечении пороговых значений, связанных с одним общим условием ухудшения функционирования, система управления поставщика анализирует и оптимальным образом коррелирует соответствующие сигналы тревоги в пределах своего домена, определяет ключевую причину проблемы и сохраняет эту информацию в регистрационном файле. Если в пределах определенного периода времени обнаруживается несколько случаев, имеющих одну и ту же корневую причину ухудшения работы, система управления поставщика готовит запись о снижении QoS для публикации всем заинтересованным пользователям (операторам или OMS). Если система управления поставщика может собрать все архивные данные обо всех контролируемых пунктах[5] на всех элементах сети в пределах управляемого ею домена, то в управлении отчетностью нет необходимости. В этом случае считается, что отчетность постоянно доступна.

9.8.1 Интерфейс ImpairmentPersistence

В пределах данной Рекомендации предполагается и требуется, чтобы объекты, имеющие данные с пороговыми значениями[6], сведенными в группы, могли быть связаны с одним и только с одним типом точки контроля.

9.8.1.1 setSlidingWindowParameters

Данная операция устанавливает параметры скользящего окна для одного, нескольких или всех контролируемых параметров NE. Эти установки применяются ко всем точкам контроля параметров. Соответственно система управления поставщика генерирует сигнал тревоги типа QoS, если в точке контроля TCA фиксируется **persistenceMinimum** раз в пределах **totConsecutiveIntvls** последовательных интервалов контроля. Эта операция является наилучшей для подобных целей.

Исключительная ситуация CommFailure возникает, если система управления поставщика не может связаться с NE, указанным в данной операции, и если значения параметров скользящего окна установлены непосредственно на сетевых ресурсах.

Эта операция используется для задания на уровне системы установок по умолчанию, если индикатор sysScopeInd имеет значение TRUE, а nEManagedEntityId является идентификатором OLT. Это означает, что значения параметров скользящего окна должны быть применены ко всем точкам контроля всех компонентов оборудования, которые в данный момент или потенциально принадлежат системе с указанным OLT. Если OLT потеряло связь с любым сетевым ресурсом до или во время выполнения этой операции, то сетевая реализация поставщика гарантирует, что установки будут применены, как только связь восстановится. Если индикатор sysScopeInd имеет значение TRUE, а nEManagedEntityId является идентификатором ONU, то установки рассматриваются как применяемые к ONU и ко всем NT, связанным с ним. Поскольку компоненты оборудования добавляются к системе с головным узлом OLT или ONU, система управления поставщика автоматически устанавливает значения этого параметра.

Если sysScopeInd имеет значение FALSE, то установки параметров скользящего окна применяются только к указанному ресурсу. Установки также применяются как к существующему, так и к потенциальным компонентам оборудования узла.

Сигнатура операции **setSlidingWindowParameters** приведена ниже:

```
void setSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean sysScopeInd)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, CommFailure);
```

---

[5] Записи архивных данных и пункты контроля определены в Рекомендации МСЭ-Т Q.834.1.

[6] Типа 21, как определено в модуле ProfileManager.

Входной параметр **nEManagedEntityId** определяет элемент сети. Входной параметр **monitoredParameterList** определяет параметры, которые будут контролироваться. Значения параметров приведены в q834_4::Q834Common::MonitoredParameter. Входной параметр **totConsecutiveIntvls** определяет суммарное значение последовательных интервалов контроля. Входной параметр **persistenceMinimum** определяет минимальное количество возникновений TCA для соответствующего monitoredParameter. Входной параметр **sysScopeInd** показывает, должны ли установки скользящего окна применяться ко всем элементам сети, связанным с начальным элементом, определенным в этой операции.

Возвращаемое значение имеет тип **void**.

9.8.1.2 setSpecificSlidingWindowParameters

Данная операция аналогична операции setSlidingWindowParameters, за исключением того, что диапазон присваивания значений установок ограничен точкой контроля, указанной в операции.

Сигнатура операции **setSpecificSlidingWindowParameters** приведена ниже:

```
void setSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in MonitoredParameterSeqType monitoredParameterList,
            in short totConsecutiveIntvls,
            in short persistenceMinimum,
            in boolean allowGlobalOverwrite)
            raises (UnknownNE, UnknownParameters, IntervalCountTooLarge,
            AccessDenied, UnknownManagedEntity, CommFailure, EquipmentFailure);
```

Входной параметр **nEManagedEntityId** определяет элемент сети, управляемый системой управления поставщика. Входной параметр **monitoringPoint** определяет указанную точку контроля для установки параметров скользящего окна контролируемого параметра. Входной параметр **monitoredParameterList** идентифицирует контролируемые параметры, значения которых определены в Q834Common::MonitoredParameter. Входной параметр **totConsecutiveIntvls** задает суммарное значение последовательно контролируемых интервалов мониторинга. Входной параметр **persistenceMinimum** определяет минимальное количество возникновений TCA для monitoredParameter. Параметр **allowGlobalOverwrite** определяет, могут ли параметры указанного скользящего окна быть перезаписаны при следующем вызове операции **setSlidingWindowParameters**.

Возвращаемое значение имеет тип **void**.

9.8.1.3 getSpecificSlidingWindowParameters

Данная операция выдает листинг точек контроля и установок их скользящих окон для указанного параметра.

Сигнатура операции **getSpecificSlidingWindowParameters** приведена ниже:

```
SWPValueSeqType getSpecificSlidingWindowParameters (
            in ManagedEntityIdType nEManagedEntityId,
            in MonitoredParameterType monitoredParameter)
            raises (UnknownNE, UnknownParameters, CommFailure);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **monitoredParameter** идентифицирует контролируемый параметр и определен в q834_4::Q834Common::MonitoredParameter.

Возвращаемое значение имеет тип **SWPValueSeqType** и содержит установки скользящего окна для указанной точки контроля.

9.8.1.4 setThreshold

Данная операция устанавливает пороговое значение, идентифицируемое профилем, для точки контроля на элементе сети.

Сигнатура операции **setThreshold** приведена ниже:

```
void setThreshold (
            in ManagedEntityIdType nEManagedEntityId,
            in ManagedEntityIdType monitoringPoint,
            in NameType thresholdDataProfileName )
            raises (UnknownNE, AccessDenied,UnknownManagedEntity,
            UnknownProfiles, InvalidAssociation, CommFailure, ProfileSuspended);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **monitoringPoint** определяет заданную точку контроля для установки параметров скользящего окна контролируемого параметра. Входной параметр **thresholdDataProfileName** выдает листинг имен профилей для пороговых значений.

Возвращаемое значение имеет тип **void**.

9.8.1.5 setThresholds

Данная операция позволяет устанавливать набор пороговых значений, пар типа monitoringPointType, на конкретном NE. Исключительная ситуация CommFailure возникает, если система управления поставщика не может связаться с NE, определенным в операции, и если пороговые значения установлены непосредственно на сетевых ресурсах.

Эта операция используется для установки в масштабах системы значений по умолчанию, если индикатор sysScopeInd имеет значение TRUE и nEManagedEntityId является идентификатором OLT. Это означает, что пороговые значения должны быть применены ко всем точкам контроля компонентов любого оборудования, принадлежащего или потенциально принадлежащего системе с оконечным узлом, имеющим имя OLT. Если OLT потеряло связь с любым сетевым ресурсом до или во время вызова операции, то реализация должна гарантировать, что установки будут применены после того, как соединение восстановится. Если индикатор sysScopeInd имеет значение TRUE и nEManagedEntityId является идентификатором ONU, то установки считаются применяемыми к ONU и ко всем NT, связанным с ним. Поскольку компоненты оборудования добавляются к системе с оконечным узлом типа OLT или ONU, система управления поставщика автоматически устанавливает эти пороговые значения.

Если sysScopeInd имеет значение FALSE, то пороговые установки применяются только к указанному ресурсу. Установки также применяются как к существующим, так и к потенциальным компонентам оборудования узла.

Сигнатура операции **setThresholds** приведена ниже:

```
void setThresholds (
                    in ManagedEntityIdType nEManagedEntityId,
                    in boolean sysScopeInd,
                    in ThresholdsSeqType thresholdsList)
                    raises (UnknownNE, UnknownProfiles, AccessDenied,
                    UnknownMonitoringPointTypes, InvalidAssociation,
                    CommFailure, ProfileSuspended );
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **thresholdList** определяет список пар Threshold Data и monitoringPointType. Входной параметр **sysScopeInd** определяет, должны ли установки скользящего окна применяться ко всем элементам сети, связанным с указанным при вызове операции.

Возвращаемое значение имеет тип **void**.

9.8.1.6 getThresholdValues

Данная операция выдает листинг точек контроля и имена установок их пороговых значений для указанного типа точек контроля.

Сигнатура операции **getThresholdValues** приведена ниже:

```
MonitoringPointThresholdsSeqType getThresholdValues (
          in ManagedEntityIdType nEManagedEntityId,
          in MonitoringKindType monitoringPointType)
          raises (UnknownNE, UnknownMonitoringPointTypes, CommFailure);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **monitoringPointType** определяет точку контроля, на которой основывается выдаваемая информация.

Возвращаемое значение имеет тип **MonitoringPointThresholdsSeqType** и содержит установки для пересечений пороговых значений.

9.8.1.7 getSystemThresholdsSetting

Данная операция выдает установленные в системе по умолчанию пороговые данные.

Сигнатура операции **getSystemThresholdsSetting** приведена ниже:

```
ThresholdsSeqType getSystemThresholdsSetting (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет систему, установки по умолчанию которой запрашиваются.

Возвращаемое значение имеет тип **ThresholdsSeqType** и содержит список системных установок по умолчанию.

9.8.1.8 getSystemSWSettings

Данная операция возвращает заданные в системе по умолчанию установки скользящего окна.

Сигнатура операции **getSystemSWSettings** приведено ниже:

```
ParameterSettingSeqType getSystemSWSettings (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (AccessDenied, UnknownManagedEntity);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет систему, для которой запрашиваются установки скользящего окна по умолчанию.

Возвращаемое значение имеет тип **ParameterSettingSeqType** и выдает список системных установок по умолчанию.

9.8.1.9 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система управления оператора не дает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, если нарушилась связь DCN между NE и системой управления поставщика.

Исключительная ситуация **EquipmentFailure** возникает, если оборудование находится в состоянии отказа и установки не могут быть применены.

Исключительная ситуация **InvalidAssociation** возникает, если данный профиль не может быть применен к точке контроля.

Исключительная ситуация **IntervalCountTooLarge** возникает, если запрошенные интервалы превышают максимум, поддерживаемый системой управления поставщика. Эта исключительная ситуация указывает максимальные допустимые интервалы контроля, поддерживаемые системой управления поставщика.

Исключительная ситуация **ProfileSuspended** возникает, если имя профиля (профилей), запрошенного при вызове операции, приостановлено для использования в пределах системы управления поставщика оператором или OMS.

Исключительная ситуация **UnknownNE** возникает, если NE, упоминаемый в запросе, неизвестен системе управления поставщика.

Исключительная ситуация **UnknownManagedEntity** возникает, если точка контроля неизвестна системе управления поставщика.

Исключительная ситуация **UnknownMonitoringPointTypes** возникает, если тип точки контроля неизвестен системе управления поставщика.

Исключительная ситуация **UnknownParameters** возникает, если данный контролируемый параметр неизвестен системе управления поставщика.

Исключительная ситуация **UnknownProfiles** возникает, если имя профиля неизвестно системе управления поставщика, и не может быть вызвано из справочника имен профилей.

9.8.2 Интерфейс ReportController

9.8.2.1 addCustomerMonitoringReporting

Данная операция добавляет тип данных архива (historydatatype), который будет контролироваться в указанной точке контроля, в ответ на жалобу пользователя, идентифицируемого с помощью serviceInstanceId, или для поддержки служб CNM.

Сигнатура операции **addCustomerMonitoringReporting** приведена ниже:

```
void addCustomerMonitoringReporting (
          in ManagedEntityIdType nEManagedEntityId,
          in ServiceInstanceIdType serviceInstanceId,
          in ManagedEntityIdType monitoringPoint,
          in GeneralizedTimeType stopTime,
          in HistoryDataType historyData,
          in short granularityPeriod)
          raises ( UnknownServiceInstance, AccessDenied, UnknownNE,
          UnknownManagedEntity, CollectionPeriodPast, CollectionLimitation,
          InvalidAssociation, UnknownHistoryDataType, CommFailure);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **serviceInstanceId** идентифицирует экземпляр пользовательской службы, связанной с точкой контроля. Входной параметр **monitoringPoint** идентифицирует указанную точку контроля для сбора архивных данных. Входной параметр **stopTime** определяет время, в которое сбор архивных данных прекратится. Если должны выдаваться отчеты, то stopTime устанавливается в "0". Если служба удалена, то выдача отчетов также автоматически завершается. Входной параметр **historyData** идентифицирует тип собираемых архивных данных; значения типов определены в q834_4::Q834Common::RecordSetType. Входной параметр **granularityPeriod** определяет длительность периода сбора в минутах. Если **stopTime** попадает в этот период, то архивные данные собираются в течение всего периода.

Возвращаемое значение имеет тип **void**.

9.8.2.2 removeCustomerMonitoringReporting

Данная операция удаляет все типы архивных данных (historydatatype), собранных ради конкретного экземпляра службы.

Сигнатура операции **removeCustomerMonitoringReporting** приведена ниже:

```
void removeCustomerMonitoringReporting (
```

```
                in ServiceInstanceIdType serviceInstanceId )
                raises ( UnknownServiceInstance, AccessDenied,
                CollectionPeriodPast, CommFailure);
```

Входной параметр **serviceInstanceId** идентифицирует экземпляр пользовательской службы, связанной с точкой контроля.

Возвращаемое значение имеет тип **void**.

9.8.2.3 selectByServiceInstance

Данная операция собирает все имеющиеся в системе управления поставщика записи, связанные с заданным serviceInstanceId. В возвращаемом списке нет избыточных записей. Система управления поставщика сканирует все найденные наборы данных.

Сигнатура операции **selectByServiceInstance** приведена ниже:

```
RecordsSeqType selectByServiceInstance (
                in ServiceInstanceIdType serviceInstanceId,
                in GeneralizedTimeType intervalStartTime,
                in GeneralizedTimeType intervalEndTime)
                raises ( UnknownServiceInstance, AccessDenied);
```

Входной параметр **serviceInstanceId** определяет экземпляр службы пользователя. Входной параметр **intervalStartTime** фильтрует выборку записей архивных данных до достижения periodEndTime. Входной параметр **intervalEndTime** фильтрует выборку записей архивных данных после periodEndTime.

Возвращаемое значение имеет тип **RecordsSeqType** и содержит список записей архивных данных, связанных с мониторингом работы, осуществляемым для данного экземпляра службы.

9.8.2.4 displayActiveReporting

Данная операция получает все точки контроля, для которых в текущий момент времени собираются архивные данные по тому или иному serviceInstanceId.

Сигнатура операции **displayActiveReporting** приведена ниже:

```
MonitoringPointSeqType displayActiveReporting (
                in ServiceInstanceIdType serviceInstanceId)
                raises ( UnknownServiceInstance, AccessDenied);
```

Входной параметр **serviceInstanceId** определяет экземпляр пользовательской службы.

Возвращаемое значение имеет тип **MonitoringPointSeqType** и содержит листинг всех активно выдающих информацию точек контроля.

9.8.2.5 addNewMonitoringReporting

Данная операция добавляет тип архивных данных, которые должны контролироваться в указанной точке контроля.

Сигнатура операции **addNewMonitoringReporting** приведена ниже:

```
void addNewMonitoringReporting (
                in ManagedEntityIdType nEManagedEntityId,
                in ManagedEntityIdType monitoringPoint,
                in GeneralizedTimeType stopTime,
                in HistoryDataType historyData,
                in short granularityPeriod)
                raises ( AccessDenied, UnknownNE, UnknownManagedEntity,
                CollectionPeriodPast, CollectionLimitation, InvalidAssociation,
                UnknownHistoryDataType, CommFailure);
```

Входной параметр **nEManagedEntityId** уникальным образом определяет элемент сети, управляемый системой управления поставщика. Входной параметр **monitoringPoint** идентифицирует указанную точку контроля для сбора архивных данных. Входной параметр **stopTime** определяет момент

времени, когда сбор архивных данных прекратится. Если установлен режим формирования отчетов, то stopTime устанавливается в "0". При удалении managedEntity (точки контроля) формирование отчетов также автоматически прекращается. Входной параметр **historyData** идентифицирует тип собираемых архивных данных; значения типов определены в q834_4::Q834Common::RecordSetType. Входной параметр **granularityPeriod** определяет период сбора в минутах. Если **stopTime** попадает в этот период, то архивные данные продолжают собираться в течение всего периода.

Возвращаемое значение имеет тип **void**.

9.8.2.6 selectByMonitoringPoint

Данная операция получает все записи, имеющиеся в системе управления поставщика и относящиеся к заданной точке контроля. Возвращаемый список не содержит избыточных записей.

Сигнатура операции **selectByMonitoringPoint** приведена ниже:

```
RecordsSeqType selectByMonitoringPoint (
               in ManagedEntityIdType monitoringPoint,
               in GeneralizedTimeType intervalStartTime,
               in GeneralizedTimeType intervalEndTime)
               raises ( UnknownManagedEntity, AccessDenied);
```

Входной параметр **monitoringPoint** определяет конкретный контроль, записи которого должны быть выданы. Входной параметр **intervalStartTime** фильтрует выборку записей архивных данных до наступления periodEndTime. Входной параметр **intervalEndTime** фильтрует выборку записей архивных данных после periodEndTime.

Возвращаемое значение имеет тип **RecordsSeqType** и содержит все записи архивных данных, сохраненные в системе управления поставщика и предоставленные точкой контроля в течение указанного периода времени.

9.8.2.7 createReportingSchedule

Данная операция добавляет запланированную выборку архивных данных, которая будет осуществляться в указанной точке контроля. Система управления поставщика будет собирать записи типа historyData для точки контроля на основе расписания. Расписание определяет periodEndTime для собираемых записей.

Сигнатура операции **createReportingSchedule** приведена ниже:

```
void createReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in ServiceInstanceIdType serviceInstance,
               in short granularityPeriod,
               in UserLabelType schedulerName)
               raises (AccessDenied, UnknownNE, UnknownManagedEntity,
               CollectionLimitation, UnknownScheduler, InvalidAssociation,
               UnknownHistoryDataType, InvalidScheduler);
```

Входной параметр **nEManagedEntityId** уникальным образом идентифицирует сетевой элемент, управляемый системой управления поставщика. Входной параметр **monitoringPoint** идентифицирует указанную точку контроля для сбора архивных данных. Входной параметр **historyData** идентифицирует тип собираемых архивных данных; значения типов определены в q834_4::Q834Common::RecordSetType. Входной параметр **serviceInstance** содержит дополнительную ссылку на копию службы. Если ссылки не требуется, то используется значение в виде пустой строки. Входной параметр **granularityPeriod** определяет период сбора в минутах. Входной параметр **schedulerName** определяет имя расписания, которое было создано ранее.

Возвращаемое значение имеет тип **void**.

9.8.2.8 modifyReportingSchedule

Данная операция модифицирует сбор архивных данных по расписанию, который будет контролироваться в указанной точке контроля. При успешном завершении изменения порядка сбора архивных данных по расписанию начнут действовать со следующего цикла.

Сигнатура операции **modifyReportingSchedule** приведена ниже:

```
void modifyReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType newSchedulerName)
               raises (AccessDenied, UnknownNE, UnknownManagedEntity,
               CollectionLimitation, UnknownScheduler, InvalidAssociation,
               UnknownHistoryDataType, InvalidScheduler);
```

Входной параметр **nEManagedEntityId** уникальным образом идентифицирует сетевой элемент, управляемый системой управления поставщика. Входной параметр **monitoringPoint** определяет указанную точку контроля для сбора архивных данных. Входной параметр **historyData** идентифицирует тип собираемых архивных данных; значения типов определены в q834_4::Q834Common::RecordSetType. Входной параметр **newSchedulerName** задает название нового расписания, которое будет применено.

Возвращаемое значение имеет тип **void**.

9.8.2.9 cancelReportingSchedule

Данная операция отменяет следующий отчет об архивных данных по расписанию, если выдача отчетов предварительно была запрошена оператором. Она не прерывает формируемый в данный момент отчет или отчет, установленный как часть взаимодействия по умолчанию между системой управления поставщика и сетевыми ресурсами.

Сигнатура операции **cancelReportingSchedule** приведена ниже:

```
void cancelReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType schedulerName)
               raises (AccessDenied, UnknownNE, UnknownManagedEntity,
               UnknownScheduler, InvalidAssociation, UnknownHistoryDataType);
```

Входной параметр **nEManagedEntityId** уникальным образом идентифицирует сетевой элемент, управляемый системой управления поставщика. Входной параметр **monitoringPoint** определяет указанную точку контроля для сбора архивных данных. Входной параметр **historyData** идентифицирует тип собираемых архивных данных; значения типов определены в q834_4::Q834Common::RecordSetType. Входной параметр **schedulerName** определяет отменяемое расписание.

Возвращаемое значение имеет тип **void**.

9.8.2.10 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система управления оператора не дает доступ к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, когда нарушается связь DCN между NE и системой управления поставщика.

Исключительная ситуация **CollectionLimitation** возникает, если система управления поставщика не может собрать данные за указанный интервал времени и период сбора из-за ограничений реализации.

Исключительная ситуация **CollectionPeriodPast** возникает, если время окончания периода меньше текущего.

Исключительная ситуация **InvalidAssociation** возникает, если данный профиль не может быть применен к monitoringPoint.

Исключительная ситуация **InvalidScheduler** возникает, если данное расписание неприменимо для использования в этой операции или устарело.

Исключительная ситуация **UnknownNE** возникает, если NE, упоминаемый в запросе, неизвестен системе управления поставщика.

Исключительная ситуация **UnknownHistoryDataType** возникает, если тип архивных данных неизвестен системе управления поставщика.

Исключительная ситуация **UnknownManagedEntity** возникает, если точка контроля неизвестна системе управления поставщика.

Исключительная ситуация **UnknownScheduler** возникает, если указанное расписание неизвестно системе управления поставщика.

Исключительная ситуация **UnknownServiceInstance** возникает, если экземпляр службы неизвестен системе управления поставщика.

9.9 Модуль ProfileManager

Данный модуль состоит из трех интерфейсов: ProfileConsumer, ProfileUsageMgr и ProfileRetriever. ProfileConsumer определяет содержание уведомления о событии при создании профиля в хранилище профилей объектов. ProfileUsageMgr поддерживает операции по управлению установками профиля, кэшируемыми в системе управления поставщика. ProfileRetriever позволяет системе управления поставщика получать значения установок профиля.

9.9.1 Интерфейс ProfileConsumer

Целью данного интерфейса является объявление о существовании новых установок профиля и транспортировка этих установок в систему управления поставщика. Данный интерфейс не имеет операций. Однако он обеспечивает отображение фиксированного заголовка, а также фильтруемых данных для структурированного объекта "событие", который используется для передачи информации о событии по каналу событий службы уведомлений OMG.

В фиксированном заголовке поле **domain_type** установлено в "telecommunications", поле **type_name** установлено в "ProfileEvent", а поле **event_name** – в постоянную строку, имеющую значение "ProfileCreation", как определено в интерфейсе.

Отображение фильтруемых данных следует стратегии использования идентификатора в форме строковой константы для компонента фильтруемых данных, за которым следует значение этого элемента данных. Кроме того, элементы фильтруемых данных перечисляются в определенном порядке.

Порядок фильтруемых элементов следующий: ProfileName, ProfileType, EventTime, ProfileAttributeValues и NotificationIdentifier. Значение ProfileName имеет синтаксис типа NameType. Значение ProfileType имеет синтаксис "короткое число без знака" и позволяет пользователю (системе управления поставщика) определять тип создаваемого профиля и разбирать значения атрибутов, найденные в дальнейшем в profileStruct. Синтаксис ProfileAttributeValues – profileStruct. EventTime имеет синтаксис типа GeneralizedTimeType и представляет собой время, когда был создан профиль. NotificationIdentifier имеет синтаксис типа NotificationIdentifierType. Этот идентификатор задает уникальную метку событию создания профиля и увеличивается при создании каждого нового профиля.

9.9.2 Интерфейс ProfileUsageMgr

9.9.2.1 reName

Данная операция обеспечивает возможность переименования профиля.

Сигнатура операции **reName** приведена ниже:

```
void reName (
            in NameType oldProfileName,
            in NameType newProfileName)
            raises (UnknownProfiles, AccessDenied, DuplicateProfileName);
```

Входной параметр **oldProfileName** – старое имя профиля, которое должно быть заменено. Входной параметр **newProfileName** – новое имя.

Возвращаемое значение имеет тип **void**.

9.9.2.2 inUse

Данная операция возвращает булевское значение, определяющее, используется ли профиль.

Сигнатура операции **inUse** приведена ниже:

```
boolean inUse (
            in NameType profileName)
            raises (UnknownProfiles, AccessDenied);
```

Входной параметр **profileName** задает имя профиля, проверяемого на использование другим участником.

Возвращаемое значение имеет тип **boolean**.

9.9.2.3 suspendUse

Данная операция приостанавливает использование профиля.

Сигнатура операции **suspendUse** приведена ниже:

```
void suspendUse (
              in NameType profileName)
              raises (UnknownProfiles, AccessDenied);
```

Входной параметр **profileName** – имя приостанавливаемого профиля.

Возвращаемое значение имеет тип **void**.

9.9.2.4 resumeUse

Данная операция возобновляет использования профиля.

Сигнатура операции **resumeUse** приведена ниже:

```
void resumeUse (
          in NameType profileName)
          raises (UnknownProfiles, AccessDenied);
```

Входной параметр **profileName** именует профиль, значения которого должны быть снова доступны для использования системой управления поставщика.

Возвращаемое значение имеет тип **void**.

9.9.2.5 deleteProfile

Операция deleteProfile обеспечивает возможность удаления неиспользуемого профиля.

Сигнатура операции **deleteProfile** приведена ниже:

```
void deleteProfile (in NameType profileName) raises (UnknownProfiles,
AccessDenied, ProfileInUse);
```

Входной параметр **profileName** – имя удаляемого профиля.

Возвращаемое значение имеет тип **void**.

9.9.2.6 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если OMS не дает доступа к запрошенной операции.

Исключительная ситуация **DuplicateProfileName** возникает, если обнаружено дублирующееся имя профиля.

Исключительная ситуация **ProfileInUse** возникает при удалении профиля и обнаруживается, когда профиль используется другим участником.

Исключительная ситуация **UnknownProfiles** возникает, если заданное имя профиля неизвестно системе управления поставщика и не может быть извлечено из хранилища профилей объектов.

9.9.3 Интерфейс ProfileRetriever

9.9.3.1 retrieve

Данная операция обеспечивает возможность выдачи профиля. Она используется системой управления поставщика для вызова профиля из системы управления оператора.

Сигнатура операции **retrieve** приведена ниже:

```
ProfileInfoType retrieve (
              in NameType profileName)
              raises (UnknownProfiles);
```

Входной параметр **profileName** – имя вызываемого профиля.

Возвращаемое значение имеет тип **ProfileInfoType** и содержит значения атрибутов указанного профиля.

9.9.3.2 Исключительные ситуации

Исключительная ситуация **UnknownProfiles** возникает, если имя заданного профиля неизвестно системе управления поставщика и не может быть извлечено из хранилища профилей объектов.

9.10 Модуль Registrar

Данный модуль поддерживает процесс включения нового NE в юрисдикцию системы управления поставщика. Он поддерживает как первоначальную установку, так и замену оборудования, мотивированную необходимостью технического обслуживания или обновления службы. Он однозначно поддерживает функцию ранжирования, используемую для измерения круговой задержки между OLT и каждым ONU или ONT и установки механизмов безопасности (алгоритм перемешивания ключа), и таймирование связи, включая автоматическую установку встраиваемого рабочего канала.

В информационной модели управления, поддерживаемой системой управления поставщика, в результате регистрации автоматически создается набор управляемых объектов. В зависимости от типа оборудования автоматически создаются копии OLT, ONT, ONU, APONLink, APONTTP, APONTrail, APONNetworkCTP, APONNetworkTTP, APONLinkConnection, tcAdapterF, vpLinkConnectionF, vpTopologicalLinkF и physicalPathTPF. В зависимости от реализации поставщика а также от того, что физически установлено, также автоматически создаются и другие управляемые объекты.

9.10.1 Интерфейс NERegistrar

9.10.1.1 registerNE

Данная операция регистрирует NE с конкретной копией системы управления поставщика. NE должен быть установлен в этой точке. Идентификатор управляемого объекта NE возвращается для индикации того, что система управления поставщика может выполнить по крайней мере одну элементарную функцию управления

Сигнатура операции **registerNE** приведена ниже:

```
ManagedEntityIdType registerNE (
                    in DCNAddressType nEDCNAddress,
                    in UserLabelType nEUserLabel,
                    in AdministrationDomainType administrationDomain)
                    raises (AccessDenied, DCNTimeout, AddressLabelMismatch,
                    DuplicateUserLabel, TooManyNEs, InvalidDCNAddress,
                    DeniedAccess, InvalidUserLabelSyntax);
```

Входной параметр **nEDCNAddress** определяет адрес DCN, используемый для доступа к NE. NE должен быть уже сконфигурирован с этим адресом. Входной параметр **nEUserLabel** задает определенную оператором метку NE. NE должен быть уже снабжен меткой пользователя. Входной параметр **administrationDomain** определяет домен управления, к которому присоединен NE.

Возвращаемое значение имеет тип **ManagedEntityIdType** и идентифицирует вновь зарегистрированный NE, если данная операция завершается успешно.

9.10.1.2 modifyNEDCNAddress

Данная операция изменяет адрес DCN зарегистрированного NE. Во время выполнения операции может быть временно потеряна связь между системой управления поставщика и NE. Новый адрес DCN должен начать действовать немедленно.

Сигнатура операции **modifyNEDCNAddress** приведена ниже:

```
void modifyNEDCNAddress (
       in ManagedEntityIdType nEManagedEntityId,
       in DCNAddressType newNEDCNAddress)
       raises(AccessDenied, DeniedAccess, AddressLabelMismatch,
       DCNTimeout, CommFailure, UnknownNE, InvalidDCNAddress, BackupInProgress);
```

Входной параметр **nEManagedEntityId** определяет NE, к которому применяется данная операция. Входной параметр **newDCNAddress**определяет новый адрес DCN NE.

Возвращаемое значение имеет тип **void**.

9.10.1.3 rangeONTorONU

Операция ранжирует ONT или ONU, используя серийный номер интерфейса PON на NE. Если ранжируемый NE еще не создан, то перед выполнением ранжирования будет вызвана операция buildNode. После успешного завершения ранжирования инициализируется синхронизация NE. Она обновит систему управления поставщика конфигурационными данными вновь ранжированного NE.

Сигнатура операции **rangeONTorONU** приведена ниже:

```
ManagedEntityIdType rangeONTorONU(
          in ManagedEntityIdType oLTManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
          in ManagedEntityIdType port)
          raises(AccessDenied, CommFailure, EquipmentFailure, UnknownNE,
```

```
UnknownPort, MaxSubtendingNodesExceeded, InsufficientPONBW,
InvalidSerialNumSyntax, APONLayerFailure, DuplicateUserLabel,
InvalidUserLabelSyntax, BackupInProgress, SynchInProgress );
```

Входной параметр **oLTManagedEntityId** идентифицирует OLT, к которому применяется эта операция. Входной параметр **nEUserLabel** – метка, присваиваемая ранжируемому NE. Входной параметр **serialNum** определяет серийный номер NE. Входной параметр **port** определяет указанный порт PON на OLT, где должен ранжироваться NE.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор управляемого объекта для ранжируемого NE.

9.10.1.4 rangeReplacementNE

Данная операция ранжирует замену ONT или ONU с использованием серийного номера заменяемого оборудования. Вся имеющаяся служебная информация соединения автоматически присваивается заменяемому NE. Сценарии замены возникают в результате деятельности по техничесокму обслуживанию для имеющихся пользователей и служб. Старая и новая пользовательские метки NE могут быть одинаковыми. За пределами IF1 поставщик и оператор должны прийти к пониманию того, какой тип оборудования может заменить любое имеющееся оборудование.

Сигнатура операции **rangeReplacementNE** приведена ниже:

```
ManagedEntityIdType rangeReplacementNE(
              in ManagedEntityIdType oldNEManagedEntityId,
              in UserLabelType newNEUserLabel,
              in SerialNumType replacementSerialNum)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidSerialNumSyntax, APONLayerFailure, EquipmentFailure,
              InvalidUserLabelSyntax, HWServicesMismatch, DuplicateUserLabel,
              BackupInProgress, SynchInProgress );
```

Входной параметр **oldNEManagedEntityId** идентифицирует заменяемый NE. Входной параметр **newNEUserLabel** идентифицирует метку для нового ранжируемого ONT или ONU. Входной параметр **replacementSerialNum** определяет серийный номер NE.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор управляемого объекта нового ранжируемого NE.

9.10.1.5 rangeUpgradeNE

Данная операция ранжирует замену NE в целях обновления оборудования. Вся имеющаяся служебная информация соединения автоматически присваивается заменяемому NE. Кроме того, предполагается, что замена NE была предварительно подготовлена (с помощью операции buildNode) и что были обеспечены любые новые служебные соединения или был зарезервирован необходимый диапазон. Новая пользовательская метка NE может быть такой же, как и старая. За пределами IF1 поставщик и оператор должны прийти к пониманию того, какой тип оборудования может заменить любое имеющееся оборудование.

Сигнатура операции **rangeUpgradeNE** приведена ниже:

```
ManagedEntityIdType rangeUpgradeNE(
              in ManagedEntityIdType oldNEManagedEntityId,
              in ManagedEntityIdType newNEManagedEntityId,
              in UserLabelType newNEUserLabel,
              in SerialNumType newNESerialNum )
              raises (AccessDenied, CommFailure, APONLayerFailure,
              EquipmentFailure, InvalidUserLabelSyntax, DuplicateUserLabel,
              UnknownNE, HWServicesMismatch, InsufficientPONBW,
              BackupInProgress, SynchInProgress );
```

Входной параметр **oldNEManagedEntityId** определяет заменяемый NE. В этом случае типом заменяемого NE является либо ONT, либо ONU. Входной параметр **newNEManagedEntityId** определяет предварительно подготовленную замену. Вся информация о новых служебных соединениях или резервировании диапазонов (полос частот) доступна системе управления

поставщика посредством этого идентификатора. Входной параметр **newNEUserLabel** задает метку для нового ранжируемого NE. Старая и новая пользовательские метки ONU или ONT могут быть одинаковыми. Входной параметр **newNESerialNum** задает серийный номер, используемый при ранжировании замены.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор управляемого объекта нового ранжируемого NE.

9.10.1.6 moveONTorONU

Данная операция используется для перемещения ONT или ONU из одного PON в другой, а также для перемещения всех связанных с ними служб.

Сигнатура операции **moveONTorONU** приведена ниже:

```
ManagedEntityIdType moveONTorONU(
                    in ManagedEntityIdType oldNEManagedEntityId,
                    in ManagedEntityIdType newPONPort)
                    raises(AccessDenied, CommFailure, UnknownNE, UnknownPort,
                    APONLayerFailure,EquipmentFailure, InsufficientPONBW,
                    BackupInProgress, SynchInProgress );
```

Входной параметр **oldNEManagedEntityId** определяет заменяемый NE. Входной параметр **newPONPort** определяет новый PON, в который перемещаются службы.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор управляемого объекта нового ранжируемого NE.

9.10.1.7 getSubtendingNEList

Данная операция возвращает все нисходящие NE для данного NE.

Сигнатура операции **getSubtendingNEList** приведена ниже:

```
ManagedEntityIdSeqType getSubtendingNEList(
                    in ManagedEntityIdType nEManagedEntityId)
                    raises (UnknownNE, AccessDenied);
```

Входной параметр **nEManagedEntityId** определяет указанный NE, для которого должен быть выдан список.

Возвращаемое значение имеет тип **ManagedEntityIdSeqType** и содержит список нисходящих сетевых элементов.

9.10.1.8 nEListGet

Данная операция возвращает сетевые элементы, находящиеся под юрисдикцией системы управления поставщика.

Сигнатура операции **nEListGet** приведена ниже:

```
ManagedEntityIdSeqType nEListGet () raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **ManagedEntityIdSeqType** и содержит список сетевых элементов, управляемых системой управления поставщика.

9.10.1.9 deRegisterNE

Данная операция выводит сетевой элемент из-под юрисдикции системы управления поставщика.

Сигнатура операции **deRegisterNE** приведена ниже:

```
void deRegisterNE (
            in ManagedEntityIdType nE)
            raises (AccessDenied);
```

Входной параметр **nE** идентифицирует сетевой элемент, выводимый из-под юрисдикции системы управления поставщика.

Возвращаемое значение имеет тип **void**.

9.10.1.10 associateNE

Данная операция ассоциирует предварительно подготовленную информацию с сетевым элементом, который был установлен и автоматически открыт. В результате действий по предварительной подготовке и установке в системе управления поставщика создается управляемый объект, доступный для OMS. Эта операция объединяет два управляемых объекта в один.

Сигнатура операции **associateNE** приведена ниже:

```
ManagedEntityIdType associateNE (
                  in ManagedEntityIdType preProvisionedNE,
                  in ManagedEntityIdType discoveredNE)
                  raises (AccessDenied, UnknownManagedEntity);
```

Входной параметр **preProvisionedNE** идентифицирует информацию, связанную с предварительно подготовленным сетевым элементом. Входной параметр **discoveredNE** идентифицирует информацию, связанную с установленным сетевым элементом.

Возвращаемое значение имеет тип **ManagedEntityIdType** и содержит идентификатор объединенной информации.

9.10.1.11 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **AddressLabelMismatch** возникает, если идентифицируемый NE не имеет текущего адреса DCN, указанного в запросе.

Исключительная ситуация **APONLayerFailure** возникает, если ранжирование протокола APON между OLT и смежным узлом завершилось неудачно.

Исключительная ситуация **BackupInProgress** возникает, если выдан запрос на отмену во время выполнения резервного копирования.

Исключительная ситуация **CommFailure** возникает, если DCN между системой управления поставщика и OLT или связь между OLT и исходным ONT нарушена.

Исключительная ситуация **DCNTimeout** возникает, если канал связи через DCN между по крайней мере одним NE и системой управления поставщика настолько перегружен, что информация о текущем состоянии не может быть передана в течение определенного в системе времени синхронизации.

Исключительная ситуация **DeniedAccess** возникает, если доступ к NE запрещен в результате действия ограничений доступа.

Исключительная ситуация **DuplicateUserLabel** возникает, если метка пользователя, заданная в запросе, используется в качестве метки другого NE или plugInUnit. Другими словами, система управления поставщика ответственна за политику присваивания меток пользователя NE и подключаемым элементам в пределах своей юрисдикции.

Исключительная ситуация **EquipmentFailure** возникает, если оборудование, с которого происходит резервное копирование, находится в состоянии отказа.

Исключительная ситуация **HWServicesMismatch** возникает, если заменяемый NE не может обеспечивать указанные службы.

Исключительная ситуация **InsufficientPONBW** возникает, если ONT или ONU не могут быть ранжированы из-за недостаточного диапазона для APONLink.

Исключительная ситуация **InvalidDCNAddress** возникает, если указанный адрес DCN неверен.

Исключительная ситуация **InvalidSerialNumSyntax** возникает, если синтаксис серийного номера не соответствует правилам определения.

Исключительная ситуация **InvalidUserLabelSyntax** возникает, если метка пользователя, присвоенная ONU или ONT, нарушает синтаксис бизнес-правил, определенных оператором и реализованных в системе управления поставщика.

Исключительная ситуация **MaxSubtendingNodesExceeded** возникает, если максимальный технический номер смежных узлов для указанного интерфейса PON превышен в запросе на предоставление услуг.

Исключительная ситуация **SynchInProgress** возникает, если любая операция запрошена в то время, когда система управления поставщика производит синхронизацию с NE.

Исключительная ситуация **TooManyNEs** возникает, если система управления поставщика не может управлять еще одним OLT.

Исключительная ситуация **UnknownManagedEntity** возникает, если оборудование неизвестно системе управления поставщика.

Исключительная ситуация **UnknownNE** возникает, если OLT неизвестно системе управления поставщика.

Исключительная ситуация **UnknownPort** возникает, если указанный порт неизвестен системе управления поставщика.

9.11 Модуль ResourceAllocation

Эта служба позволяет OMS резервировать диапазон системных ресурсов для предполагаемого служебного соединения. Зарезервированный диапазон может быть удален или выдан. Интерфейс ResourceAllocator обеспечивает средства для создания, удаления или выдачи зарезервированных ресурсов. Эта операция используется прежде всего для размещения персонала при установке сетевого элемента. Когда мощности ресурсов зарезервированы, ресурс может быть использован только для службы, указанной при резервировании.

9.11.1 Интерфейс ResourceAllocator

9.11.1.1 reserveForService

Данная операция резервирует диапазон (полосу частот) для сетевого ресурса, такого как ONT, ONU или NT, установка которых не закончена. Операция используется для ONT, ONU или NT, которые работают в первоначально выделенном диапазоне, связанном с указанным OLT. Когда операция заканчивается, возвращаемое значение типа **ReservationBandwidthType** содержит счетчик диапазонов, зарезервированных для OMS.

Сигнатура операции **reserveForService** приведена ниже:

```
ReservationBandwidthType reserveForService(
                         in EndPointType endPointA,
                         in EndPointType endPointZ,
                         in NameSeqType networkCharacteristicsProfiles,
                         in ServiceInstanceIdType serviceInstanceId)
                         raises(UnknownNE, UnknownPort, UnknownProfiles,
                         InsufficientBW, MaxSubtendingNodesExceeded,
                         ConnectionCountExceeded, CommFailure, AccessDenied,
                         ProfileSuspended );
```

Входные параметры **endPointA** и **endPointZ** идентифицирует две конечные точки соединения со службой, которые будут определять ресурсы, необходимые для поддержки ожидаемого соединения с этой службой. Входной параметр **networkCharacteristicsProfiles** – это список, состоящий из полей abTrafficDescripterProfile и baTrafficDescripterProfile (в таком порядке). Входной параметр **serviceInstanceId** идентифицирует экземпляр соответствующей службы для зарезервированного диапазона.

Возвращаемое значение имеет тип **ReservationBandwidthType** и включает идентификатор резервирования и величину диапазона сетевого ресурса, который зарезервирован системой управления поставщика. Идентификатор резервирования может использоваться OMS в процессе предоставления услуг или для отмены резервирования.

9.11.1.2 cancelReservation

Данная операция используется для удаления резервирования и освобождения ресурсов из зарезервированных мощностей системы.

Сигнатура операции **cancelReservation** приведена ниже:

```
AvailableSysBandwidthSeqType cancelReservation (
                    in ReservationIdType reservationId )
                    raises (UnknownReservationId, CommFailure, AccessDenied);
```

Входной параметр **reservationId** идентифицирует имеющееся резервирование, связанное с выделенным диапазоном.

Возвращаемое значение типа **AvailableSysBandwidthSeqType** указывает текущий доступный диапазон OLT после удаления зарезервированного диапазона.

9.11.1.3 getReservationId

Данная операция используется для отображения идентификатор резервирования, связанного с идентификатором экземпляра службы, предназначенной для зарезервированного диапазона.

Сигнатура операции **getReservationId** приведена ниже:

```
ReservationIdType getReservationId (
                  in ServiceInstanceIdType serviceInstanceId)
                  raises(UnknownServiceInstance, AccessDenied);
```

Входной параметр **serviceInstanceId** используется для выдачи идентификатора существующего экземпляра службы, присвоенного во время резервирования ресурса.

Возвращаемого значение типа **ReservationIdType** связано с идентификатором данного экземпляря службы.

9.11.1.4 reportReservedResources

Данная операция используется OMS для отображения текущего зарезервированного диапазона на указанном оконечном NE (в данном случае OLT).

Сигнатура операции **reportReservedResources** приведена ниже:

```
ReservedBandwidthSeqType reportReservedResources (
                    in ManagedEntityIdType nEManagedEntityId)
                    raises (UnknownNE, AccessDenied);
```

Входной параметр **nEManagedEntityId** используется для идентификации элемента сети, для которого запрошено резервирование текущего диапазона.

Возвращаемое значение типа **ReservedBandwidthSeqType** содержит всю имеющуюся информацию о диапазоне, зарезервированном на элементе сети.

9.11.1.5 getReservations

Данная операция используется OMS для выдачи всех резервирований, связанных с данным NE.

Сигнатура операции **getReservations** приведена ниже:

```
ReservationIdSeqType getReservations(
                        in ManagedEntityIdType nEManagedEntityId)
                        raises (UnknownNE, AccessDenied);
```

Входной параметр **nEManagedEntityId** используется для идентификации NE и выдачи текущего зарезервированного диапазона.

Возвращаемое значение типа **ReservationIdSeqType** содержит идентификатор всех резервирований, имеющихся в данный момент на OLT.

9.11.1.6 cancelAllRemainingReservations

Данная операция используется для отмены всех оставшихся резервирований мощностей, связанных с тем или иным сетевым элементом. Сетевой элемент может иметь диапазон, который либо присвоен, либо зарезервирован, либо доступен для служебных соединений. Эта операция только превращает зарезервированные ресурсы в доступные.

Сигнатура операции **cancelAllRemainingReservations** приведена ниже:

```
AvailableSysBandwidthSeqType cancelAllRemainingReservations(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, CommFailure, AccessDenied);
```

Входной параметр **nEManagedEntityId** используется для идентификации элемента сети.

Возвращаемое значение типа **AvailableSysBandwidthSeqType** указывает текущий доступный диапазон сетевого элемента после отмены зарезервированного диапазона.

9.11.1.7 getReservation

Данная операция используется для исследования происхождения резервирования мощности на NE.

Сигнатура операции **getReservation** приведена ниже:

```
ReservationInfoType getReservation (
               in ReservationIdType reservationId)
               raises (UnknownReservationId, AccessDenied);
```

Входной параметр **reservationId** для указания резервирования.

Возвращаемое значение типа **ReservationIdInfoType** содержит информацию о соединении со службой , выдаваемую как часть запроса на резервирование с указанным **reservationId**.

9.11.1.8 getAvailableSysBandwidth

Данная операция используется для определения мощности, остающейся для резервирования соединений со службами или присвоения элементу сети.

Сигнатура операции **getAvailableSysBandwidth** приведена ниже:

```
AvailableSysBandwidthSeqType getAvailableSysBandwidth (
                  in ManagedEntityIdType nEManagedEntityId)
                  raises (UnknownNE, CommFailure, AccessDenied);
```

Входной параметр **nEManagedEntityId** используется для идентификации элемента сети.

Возвращаемое значение типа **AvailableSysBandwidthSeqType** указывает текущий доступный диапазон элемента сети. Оно содержит величину диапазона, доступного для каждого порта элемента сети в зависимости от его характеристик в конкретной реализации поставщика.

9.11.1.9 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, если DCN между системой управления поставщика и OLT или связь между OLT и ONT или ONU нарушена.

Исключительная ситуация **ConnectionCountExceeded** возникает, если максимальное количество соединений для OLT или порта PON превышено в результате данного запроса на предоставление услуг.

Исключительная ситуация **InsufficientBW** возникает, если диапазон недостаточен для запрашиваемой службы.

Исключительная ситуация **MaxSubtendingNodesExceeded** возникает, если максимальный технический номер связанного узла для указанного интерфейса PON превышен в результате данного запроса на предоставление услуг.

Исключительная ситуация **ProfileSuspended** возникает, если профиль (профили), указанный в вызове операции, приостановлен для использования в системе управления поставщика оператором или OMS.

Исключительная ситуация **UnknownNE** возникает, если OLT неизвестно системе управления поставщика.

Исключительная ситуация **UnknownPort** возникает, если указанный порт неизвестен системе управления поставщика.

Исключительная ситуация **UnknownProfiles** возникает, если выданное имя профиля неизвестно системе управления поставщика и не может быть извлечено из хранилища профилей объектов.

Исключительная ситуация **UnknownReservationId** возникает, если система управления поставщика не распознает данный идентификатор резервирования.

Исключительная ситуация **UnknownServiceInstance** возникает, если экземпляры службы неизвестен системе управления поставщика.

9.12 Модуль SchedulerManagement

Данная служба используется для обеспечения интерфейсов OMS, управляющих расписаниями, которые используются для вызова различных действий. Когда расписание создано, OMS может инициировать запросы на планирование действий, таких как выгрузка NE MIB, пересылка больших объемов данных, тестирование или загрузка программ посредством обращения к расписанию. Расписания определяются потребностями рабочего окружения. Предполагается, что не должно быть расписаний, названных или установленных автоматически при инициализации системы управления поставщика. Ссылки на указанное расписание всегда делаются посредством метки пользователя. Для описания расписания используется матрица триггеров. Значения матрицы интерпретируются на основе значения HourlyDailyWeeklyMonthlyInd.

9.12.1 Интерфейс SchedulerMgr

9.12.1.1 makeScheduler

Данная операция создает новый объект расписания. OMS может затем связать действия с этим объектом путем указания ссылки на его имя – метку пользователя.

Сигнатура операции **makeScheduler** приведена ниже:

```
void makeScheduler (
            in UserLabelType schedulerName,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime,
            in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
            in TriggerTimeMatrixSeqType matrix)
            raises (InvalidStartTime, InvalidStopTime, DuplicateUserLabel,
            MatrixSchedulerTypeMismatch, AccessDenied, InvalidTrigger);
```

Входной параметр **schedulerName** идентифицирует расписание, на которое могут ссылаться другие действия. Входные параметры **startTime** и **stopTime** идентифицируют временной диапазон, в пределах которого возможно планирование действий. Входной параметр **hourlyDailyWeeklyMonthlyInd** показывает частоту (каждый час, ежедневно, еженедельно или ежемесячно), с которой должно срабатывать расписание. Входной параметр **matrix** обеспечивает конкретную информацию для вызова расписания. На него оказывает влияние значение параметра **hourlyDailyWeeklyMonthlyInd**.

Ниже приводится таблица 4, показывающая зависимость параметра **hourlyDailyWeeklyMonthlyInd** и **матрицы**:

Таблица 4/Q.834.4 – Детали матрицы

Значение hourlyDailyWeeklyMonthlyInd	Значение матрицы
Hourly (каждый час)	time – любое значение от 0 до 3600[7] dayOfWeek – должно быть 'unspecified' ("не определено") dayOfMonth – должно быть 0
Daily (ежедневно)	time – любое значение от 0 до 86 400[8] dayOfWeek – должно быть 'unspecified' dayOfMonth – должно быть 0
Weekly (еженедельно)	Time – любое значение от 0 до 86 400 dayOfWeek – отличное от 'unspecified' dayOfMonth – должно быть 0
Monthly (ежемесячно)	Time – любое значение от 0 до 86 400 dayOfWeek –должно быть 'unspecified' dayOfMonth – отличное от 0

Возвращаемое значение имеет тип **void**.

9.12.1.2 suspendScheduler

Данная операция используется для приостановки расписания. Она в основном меняет значение параметра administrativeState с "не блокировано" на "блокировано". Изменение начнет действовать со следующего цикла расписания. Данная операция вызывает приостановку всех связанных с расписанием действий.

Сигнатура операции **suspendScheduler** приведена ниже:

```
void suspendScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied);
```

Входной параметр **schedulerName** используется для идентификации приостанавливаемого расписания.

Возвращаемое значение имеет тип **void**.

9.12.1.3 resumeScheduler

Данная операция используется для возобновления приостановленного расписания. Она в основном меняет значение параметра administrativeState с "блокировано" на "не блокировано". Изменение начнет действовать со следующего цикла расписания. Данная операция вызывает возобновление всех связанных с расписанием действий.

Сигнатура операции **resumeScheduler** приведена ниже:

```
void resumeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied );
```

Входной параметр **schedulerName** используется для идентификации возобновляемого расписания .

Возвращаемое значение имеет тип **void**.

---

[7] Количество секунд от начала часа до срабатывания триггера.

[8] Количество секунд от начала дня до срабатывания триггера, где день начинается после полуночи.

9.12.1.4 modifyTime

Данная операция используется для изменения параметров расписания **startTime** и **stopTime**. Она используется OMS для увеличения или сокращения временного диапазона, в котором действует расписание.

Сигнатура операции **modifyTime** приведена ниже:

```
void modifyTime (
            in UserLabelType schedulerName,
            in GeneralizedTimeType newStartTime,
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime, InvalidStopTime, UnknownScheduler,
            AccessDenied);
```

Входной параметр **schedulerName** используется для идентификации расписания. Входной параметр **newStartTime** содержит время, в которое расписание начинает работать, а входной параметр **newStopTime** – время, в которое оно заканчивается. Если значение newStartType меньше текущего системного времени, то возникает исключительная ситуация InvalidStartTime. Значение 0 как для **newStartTime,** так и для **newStopTime** означает, что предыдущее значение не должно меняться.

Возвращаемое значение имеет тип **void**.

9.12.1.5 changeSchedulerName

Данная операция используется для изменения имени объекта расписания. При успешном завершении операции новое имя автоматически становится доступным. Любая ссылка на это расписание должна делаться с использованием вновь присвоенного имени.

Многие действия могут быть связаны с одним и тем же именем расписания. Изменение имени расписания не влияет на действия, связанные с ним. Все действия, связанные с измененным расписанием, сохраняют свои связи с ним.

Сигнатура операции **changeSchedulerName** приведена ниже:

```
void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler, DuplicateUserLabel, AccessDenied);
```

Входной параметр **oldSchedulerName** используется для идентификации существующего расписания. Входной параметр **newSchedulerName** используется для указания нового имени, присваиваемого расписанию.

Возвращаемое значение имеет тип **void**.

9.12.1.6 modifyTriggerTimes

Данная операция используется OMS для задания нового времени срабатывания триггеров и вызова расписания[9]. Если операция завершается успешно, то любые действия, связанные с расписанием, будут, начиная со следующего его срабатывания, выполняться с нового времени.

Сигнатура операции **modifyTriggerTimes** приведена ниже:

```
void modifyTriggerTimes (
          in UserLabelType schedulerName,
          in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthlyInd,
          in TriggerTimeMatrixSeqType newMatrix)
          raises (UnknownScheduler, MatrixSchedulerTypeMismatch,
          InvalidTrigger, AccessDenied );
```

Входной параметр **schedulerName** используется для идентификации модифицируемого расписания. Входной параметр **newHourlyDailyWeeklyMonthlyInd** указывает новую частоту, с которой должно

---

[9] См. сведения о зависимости используемых переменных в описании операции makeScheduler.

срабатывать новое расписание. Входной параметр **newMatrix** используется для обеспечения информации для вызова нового расписания. См. зависимости переменных в таблице 4. Как **newDailyWeeklyMonthlyInd**, так и **newMatrix** должны быть заданы явно.

Возвращаемое значение имеет тип **void**.

9.12.1.7 removeScheduler

Данная операция используется для удаления расписания. Она разрешается, только если нет никаких действий, связанных с данным расписанием.

Сигнатура операции **removeScheduler** приведена ниже:

```
void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler, AccessDenied, ScheduleInUse );
```

Входной параметр **schedulerName** используется для идентификации удаляемого расписания.

Возвращаемое значение имеет тип **void**.

9.12.1.8 retrieveScheduler

Данная операция используется для выбора информации из расписания. Объект расписания возвращается после успешного вызова операции .

Сигнатура операции **retrieveScheduler** приведена ниже:

```
SchedulerType retrieveScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler, AccessDenied);
```

Входной параметр **schedulerName** используется для идентификации отображаемого расписания.

Возвращаемое значение типа **Scheduler** содержит следующую информацию: schedulerName; startTime; stopTime; hourlyDailyWeeklyMonthlyInd; matrix; operationalState и administrativeState.

9.12.1.9 schedulerListGet

Данная операция используется для выдачи имен всех существующих расписаний, определенных для системы управления поставщика.

Сигнатура операции **schedulerListGet** приведена ниже:

```
SchedulerSeqType schedulerListGet ()
                  raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **SchedulerSeqType** и выдает требуемый листинг.

9.12.1.10 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **DuplicateUserLabel** возникает, если метка пользователя, указанная в запросе, используется для обозначения другого расписания. Другими словами, система управления поставщика ответственна за политику присваивания меток пользователя расписаниям в пределах своей юрисдикции.

Исключительная ситуация **InvalidStartTime** возникает, если указанное время начала не совместимо с существующей матрицей времени срабатывания триггеров или со временем окончания.

Исключительная ситуация **InvalidStopTime** возникает, если указанное время окончания не совместимо с существующей матрицей времени срабатывания триггеров или со временем начала.

Исключительная ситуация **InvalidTrigger** возникает, если указанный триггер содержит значения, которые не могут быть интерпретированы планировщиком расписания.

Исключительная ситуация **MatrixSchedulerTypeMismatch** возникает, если синтаксис матрицы времени триггеров не согласован с типом указанного расписания.

Исключительная ситуация **ScheduleInUse** возникает в результате выполнения операции removeScheduler в случае, если имеются операции, связанные с данным расписанием.

Исключительная ситуация **UnknownScheduler** возникает, если имя расписания не найдено.

9.13 Модуль ServiceProvisioning

С помощью этого модуля система управления поставщика выбирает порты, устройства и диапазон для того, чтобы завершить процесс планирования, выбора и присвоения, связанный со службой.

9.13.1 Интерфейс ServiceProvisioner

9.13.1.1 provisionConnection

Данная операция обеспечивает соединения между любыми двумя конечными точками волоконно-оптической системы доступа сети BPON. Соединение может быть установлено между NNI и UNI, между двумя UNI или между двумя NNI. На рисунке 1 показаны эти конечные точки.

Сигнатура операции **provisionConnection** приведена ниже:

```
ManagedEntityIdType provisionConnection(
              in EndpointType endPointA,
              in EndpointType endPointZ,
              in NameSeqType networkCharacteristicsProfiles,
              in ServiceInstanceIdType serviceInstanceId,
              in AdministrativeStateType administrativeState)
              raises (UnknownNE, UnknownProfiles, UnknownPort, InsufficientBW,
              ConnectionCountExceeded, CommFailure, EquipmentFailure,
              ParameterViolation,AccessDenied, InsufficientPONBW,
              ProfileSuspended, ConnectionAlreadyExists);
```

Входные параметры **endPointA** и **endPointZ** идентифицируют две конечные точки запроса на соединение. Конечная точка определяется структурой данных, приведенной в таблице 5.

Таблица 5/Q.834.4 – Детали конечной точки

Имя поля	Определение	Синтаксис	Комментарии
PortId	Определяет физический порт, содержащий одну из конечных точек соединения.	ManagedEntityIdType	Предполагается, что для этого порта имеется управляемый объект типа physicalPathTP.
endPointParameters	Определяет указанные параметры экземпляра службы, способствующие объединению сетевых компонентов соединения.	любой	Структуры, которые должны быть обеспечены как часть реализации. Например, соединение ATM должно иметь параметры VPI и VCI.
serviceCharacteristicsProfile	Перечисляет ссылки на профили, характеризующие службу в конечной точке.	NameSeqType	Примеры: AAL1Profile, AAL5Profile, ATMNetworkAccess Profile, UNIProfile, CESServiceProfile, DS1Profile, DS3Profile, EthernetProfile, AAL2Profile, LESProfile, SSCSParameter Profile1, SSCSParameter Profile2, VoiceService ProfileAAL2, BridgedLAN ServiceProfile, MACBridgeService Profile. (Примечание)
ПРИМЕЧАНИЕ. – Конкретный выбор зависит от службы и характеристик оборудования. Профили перечислены по именам, приведенным в модуле Q834ProfileManager. См. пример отношений в Приложении D.			

Входной параметр **networkCharacteristicsProfiles** задает список относящихся к транспорту профилей, которые будут использоваться для предоставления услуг, включая azTrafficDescriptorProfile и zaTrafficDescriptorProfile. Профили дескриптора трафика az перечислены в листинге перед профилями za. Параметр **serviceInstanceId** определяет идентификатор экземпляра службы, который будет использоваться в качестве ключа при ссылке на сетевые ресурсы, связанные с данной службой. Входной параметр **administrativeState** показывает, может или нет соединение через подсеть передавать пользовательский трафик после выполнения данной операции.

Возвращаемое значение типа **ManagedEntityIdType** представляет собой идентификатор управляемого объекта, определяющий соединение subnetworkConnection, созданное в результате данного запроса.

9.13.1.2 provisionReservation

Данная операция обеспечивает службу между NNI в OLT и UNI в ONT или между двумя UNI на основе незавершенного резервирования. Когда соединение для службы установлено, зарезервированный диапазон и соответствующий идентификатор резервирования удаляются из системы управления поставщика, поскольку зарезервированные ресурсы присвоены.

Сигнатура операции **provisionReservation** приведена ниже:

```
ManagedEntityIdType provisionReservation(
                  in ReservationIdType reservationId,
                  in AdministrativeStateType administrativeState)
                  raises ( UnknownReservationId, AccessDenied);
```

Входной параметр **reservationId** задает идентификатор резервирования. Он указывает на всю другую информацию (такую, как идентификатор экземпляра службы и конечные точки) для связи с установленным соединением. Входной параметр **administrativeState** показывает, может или нет соединение через подсеть передавать пользовательский трафик после выполнения данной операции.

Возвращаемое значение типа **ManagedEntityIdType** – это идентификатор управляемого объекта, определяющий subnetworkConnection, созданное в результате этой операции.

9.13.1.3 deleteConnection

Данная операция разрывает существующее соединение и завершает службу.

Сигнатура операции **deleteConnection** приведена ниже:

```
void deleteConnection(
            in ManagedEntityIdType subnetworkConnectionId)
            raises (UnknownConnection, CommFailure, EquipmentFailure,
            AccessDenied);
```

Входной параметр **subnetworkConnectionId** указывает подсеть, которая была предварительно создана посредством запроса на предоставление услуг.

Возвращаемое значение имеет тип **void**.

9.13.1.4 modifyConnection

Данная операция обеспечивает возможность модифицировать существующую службу.

Сигнатура операции **modifyConnection** приведена ниже:

```
ManagedEntityIdType modifyConnection (
                   in ManagedEntityIdType subnetworkConnectionId,
                   in ManagedEntityIdType portB,
                   in NameSeqType newNetworkCharacteristicsProfiles,
                   in NameSeqType newServiceCharacteristicsProfiles)
                   raises (UnknownConnection, UnknownProfiles, InsufficientBW,
                   UnknownPort, AccessDenied, ProfileSuspended );
```

Входной параметр **subnetworkConnectionId** указывает подсеть, которая была предварительно создана посредством запроса на предоставление услуг. Входной параметр **portB** помогает уточнить направленность профилей дескриптора трафика, названных в следующем входном параметре, а также определить, где должны использоваться профили службы. Порт – это порт A или Z первоначального запроса соединения. Входной параметр **newNetworkCharacteristicsProfiles** содержит модифицированный список сетевых профилей, которые будут использоваться для предоставления услуг. В списке профили дескриптора трафика "от b к противоположной конечной точке" перечислены перед профилем дескриптора трафика "от противоположной конечной точки к b". Входной параметр **newServiceCharacteristicsProfiles** содержит новый список профилей, относящихся к службе.

Возвращаемое значение типа **ManagedEntityIdType** – это идентификатор управляемого объекта, определяющий subnetworkConnection, созданное в результате данного запроса.

9.13.1.5 suspendService

Данная операция блокирует поток пользовательского трафика через соединение подсети службы. По своему действию приостановка службы эквивалентна блокировке административного состояния соединения через подсеть.

Сигнатура операции **suspendService** приведена ниже:

```
void suspendService (
            in ServiceInstanceIdType serviceInstanceId,
            in GeneralizedTimeType startTime,
            in GeneralizedTimeType stopTime)
            raises (UnknownServiceInstance, AccessDenied, InvalidStartTime,
            InvalidStopTime);
```

Входной параметр **serviceInstanceId** определяет приостанавливаемое соединение службы. Входной параметр **startTime** определяет временную точку, когда служба должна быть приостановлена. Входной параметр **stopTime** определяет временную точку, когда служба должна быть возобновлена.

Возвращаемое значение имеет тип **void**.

9.13.1.6 resumeService

Данная операция обеспечивает поток пользовательского трафика через соединение подсети службы. Она может быть вызвана либо после приостановки службы (см. операцию выше), либо если первоначальное соединение службы сконфигурировано в неактивном состоянии. Служба возобновляется немедленно.

Сигнатура операции **resumeService** приведена ниже:

```
void resumeService (
              in ServiceInstanceIdType serviceInstanceId)
              raises (UnknownServiceInstance, AccessDenied);
```

Входной параметр **serviceInstanceId** определяет возобновляемое соединение службы.

Возвращаемое значение имеет тип **void**.

9.13.1.7 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если доступ к NE запрещен по причине безопасности.

Исключительная ситуация **CommFailure** возникает, если нарушено соединение между системой управления поставщика и NE.

Исключительная ситуация **ConnectionAlreadyExists** возникает, если уже существует соединение через подсеть с теми же конечными точками.

Исключительная ситуация **ConnectionCountExceeded** возникает, если счетчик соединений превышает допустимое количество соединений.

Исключительная ситуация **EquipmentFailure** возникает, если запрос соединения не может быть применен к установленному сетевому ресурсу из-за условий сбоя на NE.

Исключительная ситуация **InsufficientBW** возникает, если диапазон недостаточен для запрошенной службы.

Исключительная ситуация **InsufficientPONBW** возникает, если доступный диапазон PON недостаточен для предоставления запрошенных услуг.

Исключительная ситуация **InvalidStartTime** возникает, если указанное время запуска несовместимо с текущим временем или временем остановки.

Исключительная ситуация **InvalidStopTime** возникает, если указанное время остановки несовместимо с текущим временем или временем запуска.

Исключительная ситуация **ParameterViolation** возникает, если параметры конечной точки не согласуются с протокольными характеристиками порта или если значение (значения) находится вне допустимого диапазона или недопустимо дублируется.

Исключительная ситуация **ProfileSuspended** возникает, если профиль (профили), указанный при вызове, приостановлен для использования в системе управления поставщика оператором или OMS.

Исключительная ситуация **UnknownConnection** возникает, если не найдено удаляемое соединение.

Исключительная ситуация **UnknownNE** возникает, если имя OLT неизвестно системе управления поставщика.

Исключительная ситуация **UnknownPort** возникает, если идентификатор входного порта неизвестен системе управления поставщика.

Исключительная ситуация **UnknownProfiles** возникает, если имя заданного профиля неизвестно системе управления поставщика и не может быть вызвано из хранилища профилей объектов.

Исключительная ситуация **UnknownReservationId** возникает, если идентификатор резервирования неизвестен системе управления поставщика.

Исключительная ситуация **UnknownServiceInstance** возникает, если идентификатор службы неизвестен системе управления поставщика.

9.14 Модуль Synchroniser

Данный модуль управляет процессом синхронизации между системой управления поставщика и заданным NE. Запрошенный процесс синхронизации может быть сведен к заданному набору листингов текущих событий. Детали того, как определяются нарушения целостности и какие данные выдаются, зависят от реализации поставщика. Однако этот процесс приводит к удалению всех несоответствий между информацией управления, имеющейся на NE, и информационной моделью, поддерживаемой в системе управления поставщика.

Операции этого модуля могут быть вызваны только привилегированным пользователем. Процесс синхронизации – наилучшее средство в том смысле, что все несоответствия определяются и исправляются до того, как зарезервирован определенный системой период тайм-аута[10]. Процесс синхронизации может в то же время дать толчок ухудшению работы системы управления поставщика до того, как он будет завершен.

Если выбранный элемент сети – OLT, то процесс синхронизации действует в масштабах системы.

9.14.1 Интерфейс NESynchroniser

9.14.1.1 synchNE

Данная операция инициирует процесс синхронизации между системой управления поставщика и указанным NE. Операция вызывает блокировку только на время, необходимое системе управления поставщика для оценки того, что процесс может быть инициирован, и выполняет операцию синхронизации в фоновом режиме.

Сигнатура операции **synchNE** приведена ниже:

```
void synchNE(in ManagedEntityIdType nEManagedEntityId)
            raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
            BackupInProgress, SynchInProgress);
```

Входной параметр **nEManagedEntityId** идентифицирует NE, синхронизирующийся с системой управления поставщика.

---

[10] Этот период тайм-аута является предметом соглашения между оператором и поставщиком.

Возвращаемое значение имеет тип **void**.

9.14.1.2 abortSynchNE

Данная операция прерывает выполняющийся процесс синхронизации между системой управления поставщика и указанным NE. Все несоответствия между системой управления поставщика и NE, которые были разрешены перед прерыванием процесса, сохраняются.

Операция может быть вызвана только привилегированным пользователем.

Сигнатура операции **abortSynchNE** приведена ниже:

```
void   abortSynchNE(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, CommFailure, UnknownNE, EquipmentFailure,
             NoSynchInProgress);
```

Входной параметр **nEManagedEntityId** определяет NE, процесс синхронизации которого с системой управления поставщика прерывается.

Возвращаемое значение имеет тип **void**.

9.14.1.3 scheduleSynchNE

Эта операция планирует процесс синхронизации между системой управления поставщика и указанным NE. Запланированный процесс синхронизации запускается планировщиком, предварительно определенным оператором. В большинстве случаев один планировщик может быть связан с этой работой на указанном NE.

Сигнатура операции **scheduleSynchNE** приведена ниже:

```
void scheduleSynchNE(
           in ManagedEntityIdType nEManagedEntityId,
           in UserLabelType schedulerName)
           raises (AccessDenied, UnknownNE, UnknownScheduler, InvalidScheduler);
```

Входной параметр **nEManagedEntityId** определяет NE, синхронизируемый с системой управления поставщика. Входной параметр **schedulerName** идентифицирует планировщика, используемого для вызова запланированной операции синхронизации системой управления поставщика.

Возвращаемое значение имеет тип **void**.

9.14.1.4 modifyNESynchSchedule

Данная операция модифицирует расписание для синхронизации NE. Она не прерывает выполняющийся процесс синхронизации. При успешном завершении новое расписание применяется со следующего вызова процесса.

Сигнатура операции **modifyNESynch** приведена ниже:

```
void modifyNESynchSchedule(
            in ManagedEntityIdType nEManagedEntityId,
            in UserLabelType newSchedulerName)
            raises (AccessDenied, UnknownNE, UnknownScheduler,
            InvalidScheduler);
```

Входной параметр **nEManagedEntityId** идентифицирует NE, где активирован запланированный процесс синхронизации. Входной параметр **newSchedulerName** идентифицирует расписание, которое будут применено.

Возвращаемое значение имеет тип **void**.

9.14.1.5 cancelScheduledSynchNE

Данная операция удаляет все последующие запланированные процессы синхронизации на данном NE. Она не прерывает выполняющийся процесс синхронизации.

Сигнатура операции **cancelScheduledSynchNE** приведена ниже:

```
void cancelScheduledSynchNE(
            in ManagedEntityIdType nEManagedEntityId )
            raises (AccessDenied, UnknownNE);
```

Входной параметр **nEManagedEntityId** идентифицирует NE, где активирован запланированный процесс синхронизации.

Возвращаемое значение имеет тип **void**.

9.14.1.6 synchCurrentEventListings

Данная операция инициирует синхронизацию между системой управления поставщика и указанным NE для элементов из конкретных листингов текущих событий. Система управления поставщика обычно возвращает текущие значения состояния, статус или атрибуты управления и делает это через итоговые листинги текущих событий системы. Если какой-либо автоматический системный процесс показывает, что листинг не соответствует текущим условиям системы, то листинг модифицируется (посредством удаления элемента или вставки нового элемента). Эта операция – ручная версия, выполняемая по запросу OMS. Когда она действует, исключительная ситуация SynchInProgress для любой другой операции не может быть вызвана.

Сигнатура операции **synchCurrentEventListings** приведена ниже:

```
void synchCurrentEventListings(
              in ManagedEntityIdType nEManagedEntityId,
              in CurrentListingSeqType currentListingTypeList)
              raises (AccessDenied, CommFailure, DCNTimeout, UnknownNE,
              EquipmentFailure, Timeout);
```

Входной параметр **nEManagedEntityId** определяет NE для синхронизации. Входной параметр **currentListingTypeList** определяет список текущих событий, которые должны быть синхронизованы между системой управления поставщика и указанным NE. Система управления поставщика возвращает значения указанного листинга текущих событий с NE.

Возвращаемое значение имеет тип **void**.

9.14.1.7 scheduledSynchNEListGet

Данная операция используется для выдачи имен всех NE с расписанием синхронизации.

Сигнатура операции **scheduledSynchNEListGet** приведена ниже:

```
ScheduledSynchNESeqType scheduledSynchNEListGet ()
                  raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **ScheduledSynchNESeqType** и содержит листинг требуемых NE.

9.14.1.8 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **BackupInProgress** возникает, если запрос на синхронизацию выдан во время выполнения резервного копирования.

Исключительная ситуация **CommFailure** возникает, если DCN между системой управления поставщика и OLT или связь между OLT и смежным ONT или ONU нарушена.

Исключительная ситуация **DCNTimeout** возникает, если канал связи DCN между по крайней мере одним NE и системой управления поставщика перегружен настолько, что информация о текущем состоянии или статусе не может быть передана в пределах установленного системой периода времени.

Исключительная ситуация **EquipmentFailure** возникает, если оборудование, с которого производится резервное копирование данных, находится в состоянии отказа.

Исключительная ситуация **InvalidScheduler** возникает, если заданное расписание не пригодно для использования в данной операции или устарело.

Исключительная ситуация **SynchInProgress** возникает, если операция synchNe запрошена в то время, когда выполняется другая операция synchNe или scheduledSynchNe.

Исключительная ситуация **SynchNotScheduled** возникает, если в указанное время для NE нет запланированной синхронизации.

Исключительная ситуация **UnknownNE** возникает, если NE, упомянутый в запросе, неизвестен системе управления поставщика.

Исключительная ситуация **UnknownScheduler** возникает, если расписание с данным именем не найдено.

Исключительная ситуация **Timeout** возникает, если указанный процесс превысил заданный в системе по умолчанию период тайм-аута.

Исключительная ситуация **NoSynchInProgress** возникает, если нет выполняющегося процесса синхронизации.

9.15 Тестовый модуль

Данная служба используется для обеспечения интерфейсов для оператора или OMS, позволяющих выполнять непосредственно запускаемые или запланированные тестовые процедуры. Тесты, такие как циклическое тестирование ячеек ATM OAM, интерфейсные циклические установки пользовательских карт или сетевых карт OLT и проверки непрерывности ATM, задаются с использованием этой службы. Тесты могут быть либо запланированными, либо запускаемыми вручную в результате обнаруженных сбоев или жалоб пользователей на качество услуг. Интерфейс TestActionPerformer обеспечивает операции, необходимые для выполнения тестов на сетевых ресурсах.

9.15.1 Интерфейс TestActionPerformer

9.15.1.1 aTMLoopback

Данная операция используется для вызова циклического теста ATM OAM. Циклический тест ATM является двунаправленным.

Сигнатура операции **aTMLoopback** приведена ниже:

```
AggregateATMLoopbackResultSeqType aTMLoopback (
                    in UserIdType testRequestorId,
                    in ManagedEntityIdType ctp,
                    in ATMLoopbackInfoType aTMLoopbackInfo,
                    in TestIterationNumType testIterationNum,
                    in ServiceInstanceIdType serviceInstanceId)
                    raises (AccessDenied, CommFailure, UnknownManagedEntity,
                    NotAvailableForTest, InvalidLocationId, InvalidDirection);
```

Входной параметр **testRequestorId** используется для идентификации инициатора циклического теста ATM. Входной параметр **ctp** используется для уникальной идентификации CTP при заведении точки входа в циклически закольцованную ячейку. Входной параметр **aTMLoopbackInfo** используется для предоставления конкретной тестовой информации ATM; LoopbackLocationId – это часть aTMLoopbackInfo. Входной параметр **testIterationNum** идентифицирует номер итерации циклического теста ATM. Входной параметр **serviceInstanceId** идентифицирует возможный экземпляр службы, связанный с запросом циклического теста.

Когда указывается параметр aTMLoopbackInfo, единственными допустимыми значениями направленности являются только "выход" или "вход". Направленность задается как часть **aTMLoopbackInfo**.

LoopbackLocationId определяет точку (точки) вдоль виртуального соединения, где должен иметь место цикл. Значение по умолчанию "все единицы" используется передатчиком для индикации конечной точки. Если segmentCellInd установлен в 'false', то LoopbackLocationId должен иметь

значение по умолчанию. Если LoopbackLocationId не задан, то предполагается, что имеется назначенная конечная точка сегмента для потока, связанного со входной ctp .

Для ctp все нули означают циклический запрос, направленный всем точкам соединения, имеющим LoopbackLocationId. Все единицы означают циклический запрос, направленный конечной точке (конечной точке сегмента или соединения). 'x6A'H означает отсутствие назначенных CP для цикла, и поэтому цикл не должен выполняться. Все другие значения LoopbackLocationId означают циклический запрос, направленный в указанное параметром LoopbackLocationId место.

Возвращаемое значение имеет тип **AggregateATMLoopbackResultSeqType** и содержит информацию о тесте, а именно loopbackingLLID, responseTime в микросекундах и признак успеха/неудачи для каждой итерации.

9.15.1.2 initializeContinuityCheck

Данная операция используется для подготовки к проверке непрерывности ATM. Создание среды для теста непрерывности не обязательно инициирует тест, но планирует тест, который будет инициализирован. После успешного создания проверки непрерывности система возвратит уникальный идентификатор, который используется для идентификации теста.

Операция относится только к установке теста. Когда тест выполнен, то в случае его неудачного завершения будет сформирован сигнал тревоги.

Сигнатура операции **initializeContinuityCheck** приведена ниже:

```
CCSetUpIdType initializeContinuityCheck(
                  in UserIdType testRequestorId,
                  in ManagedEntityIdType sourceCtp,
                  in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
                  in GeneralizedTimeType stopTime,
                  in ServiceInstanceIdType serviceInstanceId)
                  raises (AccessDenied, CommFailure, UnknownManagedEntity,
                  NotAvailableForTest, InvalidStartTime, InvalidStopTime,
                  InvalidDirection);
```

Входной параметр **testRequestorId** используется для идентификации инициатора теста. Входные параметры **sourceCtp** и **sinkCtp** используются для идентификации точки A для теста непрерывности. Входной параметр **aTMContinuityCheckInfo** используется для задания специализированной тестовой информации. Направленность должна быть установлена в 'BothDirections' ("в обоих направлениях"), а segmentCellInd – в значение 'true' для теста целостности сегмента и в значение 'false' – для теста из конца в конец. Входной параметр **stopTime** задает информацию о том, как долго должен выполняться тест. Входной параметр **serviceInstanceId** идентифицирует возможный экземпляр службы, связанный с запросом циклического теста.

Возвращаемое значение имеет тип **CCSetUpIdType,** который уникальным образом идентифицирует тест, который был установлен. CCSetUpId во время установки теста существует до тех пор, пока не наступило время stopTime, или до тех пор, пока тест не будут явно отменен с помощью операции terminateContinuityCheck.

9.15.1.3 terminateContinuityCheck

Данная операция используется для прекращения работы среды теста целостности ATM. Если тест целостности уже инициализирован (т. е. наступило время startTime), то он прекратит выполняться и будет удален.

Сигнатура операции **terminateContinuityCheck** приведена ниже:

```
void terminateContinuityCheck(
                  in CCSetUpIdType cCSetUpId)
                  raises (AccessDenied, CommFailure, UnknownTest);
```

Входной параметр **CCSetUpId** используется для идентификации завершаемого теста.

Возвращаемое значение имеет тип **void**.

9.15.1.4 scheduleResourceSelfTest

Данная операция используется для планирования теста собственных ресурсов. Эта операция используется OMS для установки выполнения теста собственных ресурсов на регулярной основе. Для выполнения данной операции необходимо наличие установленного планировщика расписания.

Сигнатура операции **scheduleResourceSelfTest** приведена ниже:

```
TestTrackingObjectIdType scheduleResourceSelfTest(
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo,
              in UserLabelType schedulerName)
              raises (AccessDenied, UnknownNE, UnknownScheduler,
              InvalidScheduler, InvalidTimeoutPeriod, InvalidTestOperations);
```

Входной параметр **testRequestorId** используется для идентификации инициатора теста. Входной параметр **targetNE** определяет сетевой элемент для выполнения самотестирования. Входной параметр **timeOutPeriod** задает максимальный период времени, в течение которого система позволит тесту работать с ресурсом. Входной параметр **specificTestInfo** используется для задания информации для самотестирования, зависящей от поставщика и относящейся к каждому типу запускаемого диагностического теста; эта информация сообщается оператору через системную документацию. Входной параметр **schedulerName** используется для ссылки на доступное расписание для данного теста.

Возвращаемое значение типа **TestTrackingObjectIdType** уникальным образом идентифицирует планируемое тестирование. Test Tracking Object существует до тех пор, пока не будет явно отменен с помощью операции terminateScheduledResourceSelfTest или не наступит конечное время планировщика.

Результаты самотестирования регистрируется. Тип данных **ResourceSelfTestResultSeqType** определяет фиксируемую часть информации.

9.15.1.5 modifyResourceSelfTestSchedule

Данная операция используется для модификации расписания регулярно проводимого тестирования собственного ресурса. При успешном выполнении инициализация самотестирования ресурса будет изменена при следующем запуске теста.

Сигнатура операции **modifyResourceSelfTestSchedule** приведена ниже:

```
void   modifyResourceSelfTestSchedule (
             in TestTrackingObjectIdType testTrackingObjectId,
             in UserLabelType newSchedulerName)
             raises (UnknownTest, UnknownScheduler, InvalidScheduler,
             AccessDenied);
```

Входной параметр **testTrackingObjectId** используется для идентификации планируемого теста. Входной параметр **testTrackingObjectId** используется для идентификации нового расписания.

Возвращаемое значение имеет тип **void**.

9.15.1.6 cancelScheduledResourceSelfTest

Данная операция используется для отмены запланированного регулярного самотестирования ресурса. При успешном завершении эта операции отменяет тест до его инициализации при очередном запуске.

Сигнатура операции **cancelScheduledResourceSelfTest** приведена ниже:

```
void  cancelScheduledResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

Входной параметр **testTrackingObjectId** используется для идентификации завершаемого регулярного теста.

Возвращаемое значение имеет тип **void**.

9.15.1.7 conductResourceSelfTest

Данная операция используется для инициализации самотестирования ресурса, вызванного системным сбоем или жалобой абонента на качество обслуживания. Результаты теста регистрируются в системе управления поставщика.

Сигнатура операции **conductResourceSelfTest** приведена ниже:

```
TestTrackingObjectIdType conductResourceSelfTest (
              in UserIdType testRequestorId,
              in ManagedEntityIdType targetNE,
              in unsigned long timeOutPeriod, //In seconds.
              in ResourceSelfTestInfoSeqType specificTestInfo)
              raises (AccessDenied, CommFailure, UnknownNE,
              InvalidTimeoutPeriod, InvalidTestOperations);
```

Входной параметр **testRequestorId** используется для идентификации инициатора теста. Входной параметр **targetNE** идентифицирует элемент сети для выполнения самотестирования. Входной параметр **timeOutPeriod** определяет период времени, в течение которого система будут пытаться инициализировать данный тест. Входной параметр **specificTestInfo** используется для задания конкретнойй информации самотестирования и содержит детали для каждого типа запускаемых диагностических тестов.

Возвращаемое значение имеет тип **TestTrackingObjectId** и обеспечивает механизм, позволяющий оператору завершать управляемое самотестирование ресурса. Если самотестирование ресурса не является управляемым, то система управления поставщика возвращает значение 0.

9.15.1.8 terminateResourceSelfTest

Данная операция завершает выполняющееся самотестирование ресурса.

Сигнатура операции **terminateResourceSelfTest** приведена ниже:

```
ResourceSelfTestResultSeqType terminateResourceSelfTest (
            in TestTrackingObjectIdType testTrackingObjectId)
            raises (UnknownTest, UncontrolledTestInProgress, AccessDenied);
```

Входной параметр **testTrackingObjectId** используется для идентификации завершаемого теста.

Возвращаемое значение типа **ResourceSelfTestResultSeqType** указывает, доступны ли промежуточные результаты тестирования.

9.15.1.9 initiateLoopback

Данная операция используется для инициализации зацикливания службы. Например, могут быть выполнены операции DS1 NearEndLineLoopback или SONET FacilityLoop.

Сигнатура операции **initiateLoopback** приведена ниже:

```
LoopbackTrackingObjectIdType  initiateLoopback (
                        in UserIdType testRequestorId,
                        in ManagedEntityIdType loopingCtp,
                        in long duration, //In minutes.
                        in DirectionalityType directionality,
                        in LoopbackTestType loopbackTest
                        in ServiceInstanceIdType serviceInstanceId)
                        raises (AccessDenied, CommFailure, UnknownManagedEntity,
                        NotAvailableForTest);
```

Входной параметр **testRequestorId** используется для идентификации инициатора теста. Входной параметр **loopingCtp** используется для идентификации CTP, на котором будет выполнено зацикливание. Входной параметр **duration** определяет период времени в секундах, в течение которого будет выполняться зацикливание. Входной параметр **directionality** определяет, будет ли зацикливание выполняться для входящего трафика, исходящего трафика или для трафика в обоих направлениях. Входной параметр **loopbackTest** используется для указания конкретного типа

циклического теста. Входной параметр **serviceInstanceId** используется для указания службы, связанной с данной операцией зацикливания.

Возвращаемое значение типа **LoopbackTrackingObjectIdType** уникальным образом идентифицирует циклический тест, который будет инициализирован. Когда циклический тест завершает свой цикл выполнения (т. е. отведенный ему интервал времени истекает), объект становится недоступным.

9.15.1.10 terminateLoopback

Данная операция используется для прекращения выполнения циклического теста.

Сигнатура операции **terminateLoopback** приведена ниже:

```
void terminateLoopback  (
                        in LoopbackTrackingObjectId loopbackTrackingObjectId)
                        raises (UnknownTest, AccessDenied);
```

Входной параметр **loopbackTrackingObjectId** используется для идентификации завершаемого циклического теста.

Возвращаемое значение имеет тип **void**.

9.15.1.11 getLoopbackInfo

Данная операция используется для выдачи циклической информации о конкретной конечной точке соединения.

Сигнатура операции **getLoopbackInfo** приведена ниже:

```
LoopbackInfoType getLoopbackInfo (
                        in ManagedEntityIdType cTP)
                        raises (UnknownManagedEntity, AccessDenied);
```

Входной параметр **cTP** используется для идентификации возможного расположения зацикливания.

Возвращаемое значение имеет тип **LoopbackInfoType** и указывает тип зацикливания и его направленность.

9.15.1.12 getLoopbackInfoByNE

Данная операция используется для выдачи места расположения и деталей каждой точки соединения на NE, находящейся в режиме зацикливания.

Сигнатура операции **getLoopbackInfoByNE** приведена ниже:

```
LoopbackInfoSeqType getLoopbackInfoByNE    (
                        in ManagedEntityIdType nEId)
                        raises (UnknownManagedEntity, AccessDenied);
```

Входной параметр **nEId** используется для идентификации сетевого ресурса.

Возвращаемое значение имеет тип **LoopbackInfoSeqType** и выдает листинг мест расположения, типов и направленностей циклических тестов.

9.15.1.13 getTestStatus

Данная операция используется для выдачи состояния выполняемого циклического теста.

Сигнатура операции **getTestStatus** приведена ниже:

```
StatusValueType getTestStatus (
                    in LoopbackTrackingObjectIdType id)
                    raises (AccessDenied, UnknownTest);
```

Входной параметр **id** определяет представляющую интерес установку теста.

Возвращаемое значение имеет тип **StatusValueType** и показывает состояние операции зацикливания.

9.15.1.14 scheduledTestNEListGet

Данная операция используется для выдачи списка всех операций тестирования, запланированных на NE.

Сигнатура операции **scheduledTestNEListGet** приведена ниже:

```
ScheduledTestNESeqType scheduledTestNEListGet ()
                    raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **ScheduledTestNESeqType** и содержит желаемый листинг.

9.15.1.15 testHistoryByManagedEntity

Данная операция используется для выдачи истории теста, связанного с управляемым объектом. Эта история должна храниться столько, сколько определено оператором.

Сигнатура операции **testHistoryByManagedEntity** приведена ниже:

```
TestHistorySeqType testHistoryByManagedEntity (
                    in ManagedEntityIdType managedEntityId)
                    raises (AccessDenied, UnknownManagedEntity);
```

Входной параметр **managedEntityId** определяет ссылку на управляемый объект.

Возвращаемое значение имеет тип **TestHistorySeqType** и выдает требуемый листинг.

9.15.1.16 testHistoryByServiceInstance

Данная операция используется для выдачи истории теста, связанного с экземпляром службы. Эта история должна храниться столько, сколько определено оператором.

Сигнатура операции **testHistoryByServiceInstance** приведена ниже:

```
TestHistorySeqType testHistoryByServiceInstance (
                    in ServiceInstanceIdType serviceInstanceId)
                    raises (AccessDenied, UnknownServiceInstance);
```

Входной параметр **serviceInstanceId** задает ссылку на экземпляр службы.

Возвращаемое значение имеет тип **TestHistorySeqType** и содержит требуемый листинг.

9.15.1.17 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если система не дает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, если DCN между системой управления поставщика и OLT или связь между OLT и исходным ONT нарушена.

Исключительная ситуация **InvalidDirection** возникает, если для данного теста задана неверная направленность.

Исключительная ситуация **InvalidLocationId** возникает, если задан неверный LLID.

Исключительная ситуация **InvalidScheduler** возникает, если заданное расписание не может быть использовано в данной операции или устарело.

Исключительная ситуация **InvalidStartTime** возникает, если заданное время начала не соответствует текущей матрице срабатывания триггеров или времени окончания.

Исключительная ситуация **InvalidStopTime** возникает, если заданное время окончания не соответствует текущей матрице срабатывания триггеров или времени начала.

Исключительная ситуация **InvalidTestOperations** возникает, если запрошенная операция самотестирования неверна.

Исключительная ситуация **InvalidTimeOutPeriod** возникает, если указанный период тайм-аута не соответствует допустимым значениям.

Исключительная ситуация **NotAvailableForTest** возникает, если sourceCTP не в состоянии задать выполнение теста целостности с помощью операции sinkCTP.

Исключительная ситуация **UncontrolledTestInProgress** возникает, если самотестирование не может быть прекращено из-за того, что тест не является управляемым.

Исключительная ситуация **UnknownNE** возникает, если NE, упомянутый в запросе, неизвестен системе управления поставщика.

Исключительная ситуация **UnknownManagedEntity** возникает, если указанный ManagedEntity неизвестен системе управления поставщика.

Исключительная ситуация **UnknownServiceInstance** возникает, если указанный экземпляр службы неизвестен системе управления поставщика.

Исключительная ситуация **UnknownScheduler** возникает, если заданное имя расписания не найдено.

Исключительная ситуация **UnknownTest** возникает, если тест, заданный с помощью trackingId, неизвестен в системе управления поставщика.

9.16 Модуль FileTransfer

Данный модуль относится к управлению передачей не в реальном времени записей, хранимых в любом краткосрочном архиве системы управления поставщика. Он поддерживает последовательное отслеживание и контроль процесса передачи файлов посредством использования объекта TransferTrackingObjectId. Система управления поставщика фиксирует успешный или неуспешный результат передачи файла. Каждый запрос на передачу сопровождается требованиями безопасности, которые разрешают системе управления поставщика соединяться с конечным сервером. Передача файла может быть запланирована заранее. Отслеживаемые объекты передачи автоматически удаляются системой управления поставщика, как только связанная с ними передача файла завершается и результаты (успешные или неуспешные) записаны в регистрационный файл.

9.16.1 Интерфейс TransferMgr

9.16.1.1 fileTransfer

Передача файла инициируется данной операцией немедленно.

Сигнатура операции **fileTransfer** приведена ниже:

```
TransferTrackingObjectIdType fileTransfer(
                    in ManagedEntityIdType recordSetId,
                    in DCNAddressType destinationServerAddr,
                    in UserIdType userId,
                    in PasswordType password,
                    in FilenameType destinationFile,
                    in boolean overwriteExistingFile)
                    raises (AccessDenied, CommFailure, UnknownRecordSet,
                    UnknownDestinationServer );
```

Входной параметр **recordSetId** определяет краткосрочный архив, из которого будут извлекаться данные для передачи файла. Входной параметр **destinationServerAddr** определяет сетевой адрес передачи данных для сервера, являющегося адресатом передачи файла. Входные параметры **userId** и **password** обеспечивают механизм входа на конечный сервер (предполагается, что необходимы соответствующие разрешения безопасности). Входной параметр **destinationFile** указывает полное расположение каталога для передаваемого файла. Наконец, параметр **overwriteExistingFile** показывает, должна или нет операция передачи разрешать перезаписывать уже существующий файл в том же конечном каталоге.

Возвращаемое значение типа **TransferTrackingObjectIdType** содержит ключ корреляции, используемый при попытке отследить состояние передачи данных не в реальном времени из краткосрочного архива в некоторый последующий момент.

9.16.1.2 scheduleFileTransfer

Данной операцией планируется будущая операция передачи файла. Все содержимое указанного краткосрочного архива извлекается для передачи файла. Не предполагается, что при успешной передаче файла данных архив должен очищаться. Вместо этого архив очищается в соответствии с соглашениями между поставщиком и оператором относительно политики сохранения архивной информации.

Сигнатура операции **scheduleFileTransfer** приведена ниже:

```
TransferTrackingObjectIdType scheduleFileTransfer (
               in ManagedEntityIdType recordSetId,
               in DCNAddressType destinationServerAddr,
               in UserIdType userId,
               in PasswordType password,
               in FilenameType destinationFile,
               in boolean overwriteExistingFile,
               in UserLabelType schedulerName)
               raises (AccessDenied,UnknownRecordSet,
               UnknownDestinationServer, UnknownScheduler, InvalidScheduler);
```

Входной параметр **recordSetId** идентифицирует краткосрочный архив, из которого будут извлечены данные для передачи файла. Входной параметр **destinationServerAddr** идентифицирует сетевой адрес передачи данных для сервера, являющегося адресатом передачи файла. Входные параметры **userId** и **password** определяют механизм входа на сервер назначения (предполагается, что нужны соответствующие разрешения безопасности). Входной параметр **destinationFile** задает полное расположение каталога для передаваемого файла. Параметр **overwriteExistingFile** показывает, разрешена ли при передаче файла перезапись уже существующего файла в том же самом конечном каталоге. Наконец, входной параметр **schedulerName** – это информация из расписания, связанного с передачей файла, на основе которой будет осуществляться данная операция.

Возвращаемое значение типа **TransferTrackingObjectIdType** содержит ключ корреляции, используемый при попытке запросить из краткосрочного архива состояние передачи данных не в реальном времени в некоторый момент в будущем.

9.16.1.3 modifyFileTransferSchedule

Данная операция используется для модификации расписания передачи файлов. При успешном завершении новое расписание начинает действовать со следующего сеанса.

Сигнатура операции **modifyFileTransferSchedule** приведена ниже:

```
void modifyFileTransferSchedule (
                    in TransferTrackingObjectIdType transferTrackingObjectId,
                    in UserLabelType newSchedulerName)
                    raises (AccessDenied, UnknownTransferProcess,
                    UnknownScheduler, InvalidScheduler);
```

Входной параметр **transferTrackingObjectId** определяет планируемый процесс передачи файлов. Входной параметр **newSchedulerName** представляет собой новую информацию расписания, связанного с передачей файла.

Возвращаемое значение имеет тип **void**.

9.16.1.4 cancelScheduledFileTransfer

Данная операция отменяет расписание передачи файлов. При успешном завершении процесс отменяется начиная со следующего планируемого запуска.

Сигнатура операции **cancelScheduledFileTransfer** приведена ниже:

```
void cancelScheduledFileTransfer (
              in TransferTrackingObjectIdType transferTrackingObjectId)
              raises (AccessDenied, UnknownTransferProcess);
```

Входной параметр **transferTrackingObjectId** определяет планируемый процесс передачи файлов.

Возвращаемое значение имеет тип **void**.

9.16.1.5 getStatus

Данная операция позволяет клиенту проверить состояние передачи до ее завершения, используя ключ.

Сигнатура операции **getStatus** приведена ниже:

```
StatusValueType getStatus (
              in TransferTrackingObjectIdType transferTrackingObjectId)
              raises (UnknownTransferProcess, AccessDenied);
```

Входной параметр **transferTrackingObjectId** определяет ключ конкретного процесса передачи файла. Оператор задает эту информацию и получает информацию о состоянии процесса передачи файла.

Возвращаемое значение имеет тип **StatusValueType** и содержит состояние процесса передачи файлов.

9.16.1.6 fileTransferHistoryListGet

Данная операция используется для выдачи списка всех завершенных процессов передачи файлов в системе управления поставщика. Этот список поддерживается системой управления поставщика в форме циклического регистрационного файла.

Сигнатура операции **fileTransferHistoryListGet** приведена ниже:

```
FileTransferHistorySeqType fileTransferHistoryListGet ()
                  raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **FileTransferHistorySeqType** и содержит требуемый листинг.

9.16.1.7 scheduledFileTransferListGet

Данная операция используется для выдачи имен всех существующих запланированных процессов передачи файлов, определенных в системе управления поставщика.

Сигнатура операции **scheduledFileTransferListGet** приведена ниже:

```
ScheduledFileTransferSeqType scheduledFileTransferListGet ()
                  raises (AccessDenied);
```

Входных параметров нет.

Возвращаемое значение имеет тип **ScheduledFileTransferSeqType** и содержит требуемый листинг.

9.16.1.8 Исключительные ситуации

Исключительная ситуация **AccessDenied** возникает, если клиент не получает доступа к объекту интерфейса.

Исключительная ситуация **CommFailure** возникает, если имеет место сбой связи между конечным сервером и системой управления поставщика.

Исключительная ситуация **UnknownDestinationServer** возникает, если указанный конечный сервер недоступен ля агента передачи.

Исключительная ситуация **UnknownRecordSet** возникает, если передаваемый файл не найден.

Исключительная ситуация **UnknownScheduler** возникает, если указанное расписание недоступно для агента передачи.

Исключительная ситуация **UnknownTransferProcess** возникает, если указанный объект **TransferTrackingObjectId** не может быть идентифицирован.

10 Заявление о соответствии

Реализация, заявляющая о соответствии любому интерфейсу, определенному в данной Рекомендации, должна полностью реализовывать поведение, относящееся ко всем операциям интерфейса, а также поведение определений из модуля q834_4::Q834Common, на которые имеются ссылки.

Приложение A

Словарь данных

В таблице A.1 приведен листинг всех элементов данных (типов данных), используемых в спецификации интерфейса и встречающихся в данной Рекомендации. Листинг включает в себя интерпретацию информации управления, синтаксис и соответствующие комментарии. Элементы данных перечислены в алфавитном порядке. Если в спецификации интерфейса одновременно присутствуют тип данных и другой тип данных, сконструированный на основе первого, то определяется только первый элемент. Интерпретация второго очевидна. В этих случаях именем последовательности является имя первоначального элемента данных с символами "Seq", вставленными перед окончанием "Type" (Типа).

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
AAL1PMHistoryDataType	Элемент данных определяет элементы в данном типе записи.	Struct	
AAL1ProfileType	Элемент данных задает значения для профиля вида AAL1.	Struct	
AAL2PMHistoryDataType	Этот элемент данных определяет элементы в данном типе записи.	struct	
AAL2ProfileType	Этот элемент данных задает значения для профиля вида AAL2.	struct	
AAL2PVCProfileType	Этот элемент данных определяет для профиля вида AAL2PVC.	struct	
AAL5PMHistoryDataType	Этот элемент данных определяет элементы в данном типе записи.	struct	
AAL5ProfileType	Этот элемент данных задает значения для профиля вида AAL5.	struct	
AALModeType	Этот элемент данных указывает, какой режим AAL используется для поддержки VCC.	enum	
ActivityLevelType	Определяет уровень разрешений доступа, присвоенный пользователю для работы.	enum	monitorOnly, allowedToExecute, noAccess

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
ActivityType	Определяет тип или категорию деятельности пользователя.	short	Определен как константы в интерфейсе q834_4:: AccessControl:: AccessControlMgr
AdministrationDomain Type	Идентификатор выдается OMS или оператором при регистрации и указывает административный домен, к которому принадлежит NE.	UserLabel Type	
AdministrativeState Type	Используется для активации (разблокировки), деактивации (блокировки) или отключения функций соответствующего управляемого объекта.	enum	Определен в Рекоменда-ции МСЭ-Т X.780.
AggregateATMLoopback ResultType	Определяет результат циклического теста ATM.	struct	
AlarmLogRecordType	Этот элемент данных определяет элементы в данном типе записи.	struct	
AlarmStatusSeqType	Этот элемент данных обеспечивает все соответствующие значения для переменных состояния сигнала тревоги.	enum	Допустимые значения для enum: AS_Under Repair, AS_Critical, AS_Major, AS_Minor, AS_AlarmOut standing
AnnouncementType	Этот элемент данных обеспечивает уведомление пользователя о разрыве связи, если не было сделано ни одной попытки звонка.	enum	
APONPMHistoryDataType	Этот элемент данных определяет элементы в данном типе записи.	struct	
AppIdType	Этот элемент данных определяет комбинации протоколов, используемых в функциях взаимодействия между голосовым шлюзом и ONT или NT.	enum	
ATMContinuityCheckInfo Type	Определяет вход для проверки целостности ATM.	struct	
ATMLoopbackInfoType	Определяет информацию для циклического теста ATM.	struct	
ATMNetworkAccess ProfileType	Профили и копия профиля типа ATMNetworkAccess	struct	
ATMOverbookingFactor Type	Этот элемент данных задает значение для избыточной регистрации ATM.	struct	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
AudioServIndType	Этот булевский элемент данных указывает, передана или нет аудиослужба, значение TRUE подразумевает присутствие данной службы.	boolean	
AutoDetectionIndType	Этот булевский элемент данных указывает, включен ли режим автоопределения скорости передачи данных.	boolean	
AvailableSysBandwidthSeqType	Это листинг распределения доступных диапазонов по портам.	Последовательность PortBand и widthType	
AvailabilityStatusSetType	Этот элемент данных содержит все соответствующие значения состояния переменной определения доступности.	Последовательность значений состояния доступности	Определен в Рекомендации МСЭ-Т X.780.
BackedUpStatusType	Этот элемент данных указывает, есть ли у управляемого объекта формирования сигналов тревоги модуль резервного копирования.	Boolean	Определен в Рекомендации МСЭ-Т X.780.
BridgedLANServiceProfileType	Этот элемент данных задает значения профиля типа "служба LAN с использованием моста".	struct	
BridgePriorityType	Этот булевский элемент данных указывает, доступны ли обучающие функции моста. TRUE означает, что доступны.	short	
BRISignallingType	Этот элемент данных выбирает, какой формат сигнализации будет использоваться для основной скорости ЦСИС.	enum	
BufferedCDVToleranceType	Этот элемент данных задает время передачи пользовательских данных, в течение которого они должны быть буферизированы промежуточным объектом CES для сдвига изменений задержек в ячейке. Интервал времени задается с шагом в 10 микросекунд.	long long	
CableLengthType	Этот элемент данных задает длину кабеля витой пары от интерфейса physicalPathTP типа "DS1" до точки пересечения DSX1 (если она используется).	long long	
CASType	Этот элемент данных выбирает, какой формат AAL1 должен использоваться. Он применяется только к структурированным интерфейсам. Для неструктурированных интерфейсов это значение, если оно имеется, должно быть установлено в значение по умолчанию.	enum	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
CASIndType	Этот булевский элемент данных указывает, включена ли при соединении сигнализация связанного канала. Значение TRUE означает, что включена.	boolean	
CBRRateType	Этот элемент данных содержит скорость службы CBR, поддерживаемой AAL.	enum	
CCSetUpIdType	Определяет уникальный Id для установки теста CC.	long long	
CDVTPCREgressType	Устойчивость к изменениям задержек в ячейке – этот параметр требуется для всех категорий служб. Он применяется для потока ABR с CLP = 0 и для потоков с CLP = 0 + 1 в остальных случаях.	long long	
CDVTPCRIngressType	Устойчивость к изменениям задержек в ячейке – этот параметр требуется для всех категорий служб. Он применяется для потока ABR с CLP = 0 и для потоков с CLP = 0 + 1 в остальных случаях.	long long	
CellLossIntegration PeriodType	Этот элемент данных содержит время в миллисекундах для периода интеграции потерь ячеек. Если ячейки теряются на этот интервал, связанный с ними промежуточный объект vcCTPF генерирует сигнал о зависании.	long long	
CESServiceProfileType	Этот элемент данных задает значения для профиля типа "Служба CES".	struct	
ClockRecoveryType	Этот элемент данных указывает, можно ли извлечь тип восстановления часов из физического интерфейса.	enum	
CMDataIndType	Этот булевский элемент данных указывает, действует ли в данном соединении канальный режим передачи данных; значение TRUE означает, что действует.	boolean	
CMMultiplierNumType	Этот элемент данных задает значение *N* в *N* × 64 кбит/с для передачи данных в канальном режиме.	short	
ConformanceDefType	Указывает тип подтверждения, как определено на форуме ATM-TM 4.0.	enum	
ControlStatusSetType	Этот элемент данных задает все возможные значения переменной управления состоянием.	Последова-тельность enum	Определен в Рекомендации МСЭ-Т X.780.

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
CorrelatedNotification Type	Этот элемент данных перечисляет ссылочные номера других уведомлений о событиях, которые имеют отношение к уведомлению о данном событии.	Последова-тельность Notification IdentifierType	
CreationModeType	Этот элемент данных указывает, как был создан набор записей.	enum	Выбор между заданным оператором и установкой в качестве части первоначальной инсталляции приложений системы управления поставщика.
CurrentListingType	Задает список листингов типов текущих событий, которые могут быть синхронизированы между системой управления поставщика и NE.	short	Значения, определенные в форме констант.
CurrentSizeType	Этот элемент данных описывает текущий размер набора записей.	unsigned long long	В тех же единицах, что и MaxSizeType.
DataRateType	Этот элемент данных задает скорость передачи данных через Ethernet-соединение. Допустимые значения: 10 Мбит/с или 100 Мбит/с.	enum	
DayOfMonthType	Указывает день месяца.	short	0 означает, что значение не определено.
DayOfWeekType	Указывает день недели в расписании.	enum	Воскресенье, Понедельник, Вторник, Среда, Четверг, Пятница, Суббота, Не определено
DCNAddressType	Задает адреса NE или системного сервера сети передачи данных оператора. Используется для маршрутизации сообщений.	string	Обычно IP-адрес. Примеры включают элементы, помеченные как softwareSource Addr, destinationServer Addr, sourceServerAddr, nEDCNAddress и newNEDCNAddress.
DefaultSSCSParameter Profile1PtrType	Этот элемент данных определяет значения по умолчанию для профиля службы, связанного с управлением каналом передачи и трафиком.	Name	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
DefaultSSCSParameter Profile2PtrType	Этот элемент данных определяет значения по умолчанию для объединенного профиля службы, связанного с каналами, несущими медиапотоки (например, каналами POTS или ЦСИС B).	Name	
DiagnosticType	Определяет тип дианостического теста.	short	Зависит от поставщика
DirectionalityType	Определяет направленность теста.	enum	Возможные значения: Egress, Ingress, BothDirections
DirectionType	Определяет направление оптический волны, связанной с портом.	enum	Значения: в одном направлении или в двух направлениях.
DownloadStatusSeqType	Этот элемент данных определяет состояние процесса загрузки программ.	struct, включающая Id конечного объекта, за которым следует состояние доставки, распределения, подтверждения и активации	
DS1EncodingType	Этот элемент данных определяет тип кодирования в линии DS1.	enum	
DS1FramingType	Этот элемент данных определяет тип применяемого кадрирования.	enum	
DS1PMHistoryDataType	Этот элемент данных задает элементы данного типа записи.	struct	
DS1ProfileType	Этот элемент данных задает значения для профиля типа DS1.	struct	
DS3ApplicationType	Этот элемент данных определяет тип сигнала DS3.	enum	
DS3PMHistoryDataType	Этот элемент данных задает элементы данного типа записи.	struct	
DS3ProfileType	Этот элемент данных задает значения для профиля типа DS3.	struct	
DS3EncodingType	Этот элемент данных определяет тип кодирования в линии DS3.	enum	
DTEDCEType	Этот элемент данных указывает используемый интерфейс записи в Ethernet: DTE или DCE.	boolean	
DTMFIndType	Этот булевский элемент данных указывает, передаются ли при соединении цифры двухтонального многочастотного набора номера; значение TRUE означает, что передаются.	boolean	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
DuplexType	Этот элемент данных указывает, используется ли полностью дуплексный (=TRUE) или полудуплексный (=FALSE) режим.	boolean	
E1PMHistoryDataType	Этот элемент данных определяет элементы записи данного типа.	struct	
E3PMHistoryDataType	Этот элемент данных определяет элементы записи данного типа.	struct	
EchoCancellationIndType	Этот элемент данных указывает, имеется ли схема подавления эха.	boolean	
ELCPIndType	Этот булевский элемент данных показывает, используется ли протокол управления эмулируемой петлей.	boolean	
EncapsulationProtocolType	Этот элемент данных определяет протокол инкапсуляции, используемый при включении в LAN моста через ATM.	short	
EncProfileIndexType	Этот элемент данных указывает конкретный предварительно определенный профиль кодирования, используемый в данном случае.	short	
EncSrcTypeType	Этот элемент данных определяет источник для формата профиля кодирования. Возможные значения включают в себя "ITU-T" ("МСЭ-Т") и "ATM Forum" ("Форум ATM"), но не ограничиваются ими	enum	
EndPointType	Определяет характеристики конечной точки соединения.	struct	
EquipmentHolderAddressType	Этот элемент данных определяет расположение оборудования для схемного пакета.	struct, компоненты которой включают номер полки (короткий) и номер порта (короткий)	
EquipmentHolderFType	Эта структура данных определяет элемент equipmentHolderF, названный в событии автоматического открытия.	struct, перечисляющая значения атрибутов для equipmentHolderFManagedEntity	
EquipmentType	Этот элемент данных определяет инвентарный элемент, названный в событии автоматического открытия.	Short	
EthernetPMHistoryDataType	Этот элемент данных определяет элементы данного типа записи.	struct	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
EthernetProfileType	Этот элемент данных определяет значения для профиля типа Ethernet.	struct	
ExternalTimeType	Этот элемент данных устанавливает локальное время, которое будет связываться с сетевым ресурсом.	Generalized TimeType	
FaxDemodIndType	Этот булевский элемент данных определяет, используется ли метод демодуляции FAX; значение TRUE показывает, что используется.	boolean	
FilenameType	Определяет полное местоположение файла. Этот файл может содержать конфигурационные данные, загруженные с NE, место, куда будут пересланы записи файла, или расположение нужного программного обеспечения на NE.	string	
FileTransferHistory Type	Этот элемент данных определяет запись, относящуюся к передаче файла, зафиксированную в системе управления поставщика.	struct	
FilterType	Этот элемент данных определяется элементом CosNotifyFilter::Filter.		
FMDataIndType	Этот булевский элемент данных указывает, используется ли в данном соединении передача данных в кадровом режиме; значение TRUE указывает на то, что используется.	boolean	
FMMaxFrameLenType	Этот элемент данных задает максимальную длину элемента данных при передаче в кадровом режиме.	long long	
ForwardDelayType	Этот элемент данных задает время (в сотых долях секунды), в течение которого мост на Ethernet-карте в ONT (в качестве члена сообщества всех мостов в локальной сети) удерживает пакет перед тем, как переслать его.	short	
ForwardErrorCorrection Type	Этот элемент данных определяет метод FEC.	enum	
FullActionType	Определяет поведение записи, если она достигает своего максимального размера.	enum	Выбор между разбиением на части и остановкой.
GeneralizedTimeType	Задает нормализованную метрику времени. Используется для того, чтобы избежать неоднозначности, связанной с временными зонами.	string	Только один разрешенный формат YYYYMMDDHHMMSS.fffZ (т. е. время по Гринвичу).

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
HelloTimeType	Этот элемент данных задает интервал времени (в сотых долях секунды) между пакетами приветствия. Это время, в течение которого мост обнаруживает свое присутствие в качестве корня или пытается стать корнем.	short	
HistoryDataType	Определяет тип записи, содержащей статистику выполнения контроля.	RecordKind Type	
HourlyDailyWeekly MonthlyIndType	Определяет цикл расписания: ежечасно, ежедневно, еженедельно или ежемесячно.	enum	Ежечасно, Ежедневно, Еженедельно, Ежемесячно
IDLCCallProcessing ProfileType	Этот элемент данных определяет значения для профиля обработки вызовов типа IDLC.	struct	
JitterBufferMaxType	Этот элемент данных определяет максимальную глубину буфера отклонений, относящегося к данной службе. Измеряется в милисекундах.	long long	
JitterTargetType	Этот элемент данных задает значение буфера отклонений. Система будет стремиться поддерживать величину буфера в заданном значении. Измеряется в милисекундах.	long long	
LANType	Этот элемент данных содержит информацию о типе используемой LAN, например ethernet, token-ring и т. д.	short	
LESProfileType	Этот элемент данных задает значения для профиля типа LES.	struct	
LocalMaxNumVCC SupportedType	Этот элемент данных определяет количество VCC, которое может поддерживаться ATM NE на этом конце интерфейса.	long long	
LocalMaxNumVCIBitsType	Этот элемент данных определяет максимальное количество распределенных битов в подполе VCI , которое может поддерживаться FSAN NE на этом конце интерфейса.	short	
LocalMaxNumVPC SupportedType	Этот элемент данных определяет количество VPC, которое может поддерживаться OLT на этом конце интерфейса.	long long	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
LocalMaxNumVPIBitsType	Этот элемент данных определяет максимальное число распределенных битов в подполе VPI , которое может поддерживаться FSAN NE на этом конце интерфейса.	short	
LoopbackCodeType	Этот элемент данных определяет тип поддерживаемого циклического внутриполосного кода.	enum	
LoopbackInfoType	Этот элемент данных определяет тип цикла, направленность и расположение.	struct	
LoopbackInfoSeqType	Задает информацию, относящуюся к условию выполнения одного цикла на сетевом ресурсе.	struct	
LoopbackLocationIdSeq Type	Задает уникальный Id расположения ctp.	Последова-тельность октетов длиной 16	
LoopbackLocCodeType	Этот элемент данных задает код, который идентифицирует уровень ATM в OAM для входящих ячеек цикла, которые должны быть зациклены в обратном направлении на этом UNI.	string	
LoopbackTestType	Этот элемент данных определяет тип установки циклического теста, которая используется или будет использоваться.	unsigned short	Возможные значения определены в PhysicalLayerLoop back interface в Q834::Common.
LoopbackTrackingObject IdType	Этот элемент данных определяет текущие условия зацикливания.	Tracking ObjectIdType	
MACBridgePortPMHistory DataType	Этот элемент данных определяет элементы записи данного типа.	struct	
MACBridgePMHistoryData Type	Этот элемент данных определяет элементы записи данного типа.	struct	
MACBridgeService ProfileType	Этот элемент данных определяет значения для профиля типа MACBridgeService.	struct	
ManagedEntityIdType	Этот элемент данных определяет уникальное имя управляемого объекта. Имя включает в себя индикацию, если оно является именем управляемого поэлементно объекта, именем объекта типа façade или просто ссылкой на структуру данных.	struct enum и тип RDNType	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
MaxAgeType	Этот элемент данных указывает максимальный возраст (в секундах) объекта в листинге типа разветвляющегося дерева. Он соответствует максимальному времени в секундах для хранения принятой информации протокола до того, как она будет сброшена.	short	
MaxBSEgressType	Максимальный размер пачки – этот параметр требуется для трафика VBR и GFR, передаваемого в реальном времени или не в реальном времени. Он применяется для потока трафика VBR.1, GFR.1 и GFR.2 с CLP = 0 + 1 и для трафика VBR.2 и VBR.3 с CLP = 0.	long long	
MaxBSIngressType	Максимальный размер пачки – этот параметр требуется для трафика VBR и GFR, передаваемого в реальном времени или не в реальном времени. Он применяется для потока трафика VBR.1, GFR.1 и GFR.2 с CLP = 0 + 1 и для трафика VBR.2 и VBR.3 с CLP = 0.	long long	
MaxCPCSSDUSizeType	Этой многозначный элемент данных определяет максимальный размер файла CPCS_SDU, который будет передаваться через соединение как во входящем (прямом), так и в исходящем (обратном) направлении передачи.	struct	
MaxCPS_SDULengthType	Этот элемент данных задает максимальную допустимую длину сервисного блока данных общего частичного подуровня (CPS SDU), который может быть передан через соединение как в восходящем, так и в нисходящем направлении передачи.	long long	
MaxFrameSizeType	Этот элемент данных обозначает максимальный допустимый размер кадра для передачи через интерфейс.	long long	
MaximumChanIdType	Этот элемент данных определяет максимальное значение Id канала, разрешенное внутри соединения.	short	
MaxNumChannelsType	Этот элемент данных задает максимальное количество каналов, которые может содержать тракт VC, связанный с промежуточным vcCTP.	short	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
MaxNumCIDsType	Этот элемент данных задает максимальное количество активных каналов в VCC.	short	
MaxPacketLengthType	Этот элемент данных определяет максимальную длину пакета.	long long	
MaxSizeType	Определяет максимальный размер набора записей.	unsigned long long	В единицах, определяемых совместным соглашением между поставщиком и оператором.
MaxSSSARSDULengthType	Этот элемент данных задает максимальную длину, допустимую для SSSAR-SDU объединенного подуровня сходимости сегментации и зависящей от службы повторной сборки.	long long	
MFR1IndType	Этот булевский элемент данных показывает, передаются ли в соединении цифры R1 двухтонального многочастотного набора номера; значение TRUE указывает, что передаются.	boolean	
MFR2IndType	Этот булевский элемент данных передаются ли в соединении цифры R2 двухтонального многочастотного набора номера; значение TRUE указывает, что передаются.	boolean	
MFSEgressType	Максимальный размер кадра – этот параметр требуется только для трафика GFR.	long long	
MFSIngressType	Максимальный размер кадра – этот параметр требуется только для трафика GFR.	long long	
MinimumChanIdType	Этот элемент данных задает минимальное значение для Id канала, разрешенное в соединении.	short	
MonitoredParameterSeq Type	Этот элемент данных определяет контролируемый параметр.	string	Контролируемые параметры определены в Q834Common:: Monitored Parameter
MonitoringKindType	Этот элемент данных определяет тип указанного контроля показателей.	string	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
MonitoringPoint ThresholdsType	Этот элемент данных содержит листинг копий контролируемых точек контроля и соответствующие пороговые данные для указанного сетевого ресурса.	Последовательность struct	
NameType	Этот элемент данных задает имя CORBA для объекта профиля.	CosNaming:: Name	
NEFSANType	Эта структура данных определяет инвентарный номер оборудования, упомянутый в событии автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта NEFSAN	
NEKindType	Этот элемент данных определяет тип сетевых элементов, которые могут создаваться.	enum	Выбор между BPONOLT, BPONONT, BPONONU и BPONNT (Примечание).
NotificationIdentifier Type	Уникальным образом идентифицирует уведомление.	long long	
NTType	Эта структура данных определяет инвентарный номер NT, называемый при событии автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта NT	
OLTType	Эта структура данных определяет инвентарный номер OLT, называемый при событии автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта OLT	
ONTType	Эта структура данных определяет инвентарный номер ONT, называемый при событии автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта ONT	
ONUType	Эта структура данных определяет инвентарный номер ONU, называемый при событии автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта ONU	
OperationalStateType	Определяет рабочее состояние (включен или отключен) управляемого объекта.	enum	Определен в Рекомендации МСЭ-Т X.780.
OpticalWaveLengthArray SeqType	Определяет длину волны (в нанометрах) и направление волны, мультиплексируемой в данном оптическом порту.	struct	
ParameterSettingSeq Type	Этот элемент данных определяет параметр контроля наряду с параметрами связанного с ним скользящего окна.	struct	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
PartiallyFilledCells Type	Этот булевский элемент данных определяет количество используемых ведущих октетов.	long long	
PasswordType	Задает установки безопасности для доступа к приложениям или серверам системы управления поставщика.	string	Пароль, поставляемый вместе с идентификатором пользователя, должен соответствовать политике паролей (passwordPolicy), обеспечиваемой системой управления поставщика.
PasswordPolicyType	Определяет политику паролей, обеспечиваемую системой управления поставщика. Она состоит из двух компонентов: UserLoginPolicyType и SessionPolicyType.	struct	
PCMEncTypeType	Этот элемент данных определяет тип кодирования ИКМ. Допустимые значения включают в себя "ИКМ-кодирование по μ-закону" и "ИКМ-кодирование по α-закону", но не ограничиваются ими.	short	
PCREgressType	Пиковая скорость ячейки – этот параметр требуется для трафика всех категорий служб. Он применяется для потока ABR с CLP = 0 и для потоков с CLP = 0 + 1 в остальных случаях.	long long	
PCRIngressType	Пиковая скорость ячейки – этот параметр требуется для трафика всех категорий служб. Он применяется для потока ABR с CLP = 0 и для потоков с CLP = 0 + 1 в остальных случаях.	long long	
PerceivedSeverityType	Определен в Рекомендации МСЭ-Т X.780.	enum	
PIDType	Этот элемент данных определяет значения медиатипов, которые могут использоваться при инкапсуляции ATM (определен в RFC 1483).	short	
PlugInUnitFType	Эта структура данных определяет инвентарный номер PlugInUnitF, называемый при возникновении события автооткрытия.	struct, перечисляющая значения атрибутов для управляемого объекта PlugInUnit.	
PlugInUnitType	Зависящее от реализации имя схемы пакета, обеспечиваемое поставщиком.	string	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
PortBandwidthSeqType	Этот элемент данных задает диапазон для определенного порта.	struct	Примеры листингов включают ReservedBandwidthSeqType и AvailableSysBandwidthSeqType
POTSSignallingType	Этот элемент данных выбирает, какой формат сигнализации будет использоваться службой POTS.	enum	
ProbableCauseType	Данные для возможной причины.	unsigned short	Значения определены в Q834Common::ProbableCause и в Рекомендации МСЭ-Т X.780
ProceduralStatusSetType	Определяет состояние выполнения незавершенных действий.	Последовательность enum	Определен в Рекомендации МСЭ-Т X.780
ProfileInfoType	Этот элемент данных определяет тип профиля и значения его атрибутов.	struct	
ProfileKindType	Этот элемент данных определяет тип профиля с данным именем.	unsigned short	Значения приведены в Q834Common.
ProtectionParameterType	Этот элемент данных описывает параметры защитного переключения, связанные с группами управляемых объектов. Параметры включают в себя отношение защитного переключения, его допустимые механизмы, возвращаемый индикатор и время ожидания для возврата в исходное состояние.	struct	
ProtectionUnitType	Этот элемент данных связывает защищаемые и защищающие сетевые ресурсы.	struct	
RASTimerType	Этот элемент данных задает время повторной сборки (в секундах) подуровня сходимости сегментации и зависящий от службы повторной сборки для Рекомендации МСЭ-Т I.366.1.	short	
RateControlIndType	Этот булевский элемент данных указывает, применяется ли при передаче через соединение управление скоростью; значение TRUE говорит о том, что применяется.	boolean	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
RecordType	Это элемент, хранимый в наборе записей.	any	Значение – структура на основе RecordKindType, определенного в Q834Common:: RecordSetType. В наборе записей может быть сохранен только один параметр RecordKind.
RecordSetIdType	ManagedEntityId набора записей.	ManagedEntity IdType	
RecordSetStatusType	Определяет текущее состояние RecordSet.	struct из currentSize Type, Operational StateType, MaxSize Type, SizeThreshold Type, filterName, Administrative StateType, RecordKind Type и RecordSet UserLabel	
RecordKindType	Определяет тип записи.	unsigned short	Значения определены в Q834Common:: RecordSetType
RecordsSeqType	Этот элемент данных задает набор записей, сгруппированных по типам.	Последова-тельность any	См. выше
RemainingPassword Validity	Определяет длительность действия пароля в количестве дней.	long	Значение ≥ 1
ReservationIdType	Идентификатор, который коррелирует диапазон, зарезервированный на PON или в системе OLT с экземпляром службы.	string	
ReservationInfoType	Этот элемент данных содержит полный отчет о резервировании и связанных с ним соединениях службы.	struct	
ReservedBandwidthSeq Type	Определяет величину диапазона, связанного с Id резервирования для порта.	Последова-тельность PortBand widthType	
ResourceSelfTestInfo Type	Задает тестовую диагностику, которая будет использоваться при самотестировании ресурсов.	short	Значения определяются реализацией поставщика.

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
ResourceSelfTestResultType	Определяет один результат для самотестирования ресурса.	struct	
ScheduledFileTransferType	Определяет процесс ждущей запланированной передачи файла для системы управления поставщика.	struct	
ScheduledSynchNEType	Этот элемент данных связывает NE с его расписанием синхронизации.	struct	
ScheduledTestNESeqType	Этот элемент данных связывает NE с запланированным тестом.	struct	
SchedulerType	Определяет копию планировщика в системе управления поставщика.	struct	
SCREgressType	Нормальная скорость ячейки – этот параметр применяется к VBR в реальном и не в реальном времени. Он применяется для потока трафика VBR.1 с CLP = 0 + 1 и для потоков VBR.2 и VBR.3. с CLP = 0.	long long	
SCRIngressType	Нормальная скорость ячейки – этот параметр применяется к VBR в реальном и не в реальном времени. Он применяется для потока трафика VBR.1 с CLP = 0 + 1 и для потоков VBR.2 и VBR.3. с CLP = 0.	long long	
SecurityAlarmLogRecordType	Этот элемент данных определяет элементы в данном типе записи.	struct	
SegmentEndpointIndType	Определяет, будет ли сконструированный vpNetworkCTP рассматривается как сегмент или конечная точка циклического теста ячейки ATM OAM.	enum	Выбор из сегмента, конечной точки или ничего
SegmentLengthType	Этот элемент данных задает длину сегмента для подуровня сходимости сегментации и зависящей от службы повторной сборки. Она находится в диапазоне от 0 до максимального значения, заданного элементом данных MaxCPS_SDULen.	long long	
SerialNumType	Задает уникальный Id оборудования, используемый в процессе выделения диапазонов, который описан в Рекомендации МСЭ-Т Q.983.1.	string	
ServiceAffectingType	Этот элемент данных указывает, влияет ли состояние отказа на службу, если оно может быть определено.	enum	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
ServiceCategoryType	Указывает категорию службы, как определено в Форуме ATM TM 4.0. Допустимые значения: CBR, rt-VBR, nrt-VBR, UBR, ABR или GFR.	enum	
ServiceCatType	Этот элемент данных указывает тип категории службы, определенный в AAL2. Допустимые значения включают в себя "Audio" и "Multirate", но не ограничиваются ими.	enum	
ServiceInstanceIdType	Задаваемая оператором метка, связываемая системой управления поставщика с соединениями.	string	
ServiceOutageRecord Type	Этот элемент данных определяет элементы в данном типе записи.	struct	
SessionPolicyType	Определяет правила управления GUI сессиями клиента, взаимосвязанными с системой управления поставщика.	struct	
SizeThresholdType	Определяет порог в наборе записей, при достижении которого возникает сигнал тревоги.	unsigned short	Значение интерпретируется как целое число процентов (0–100%) от MaxSize.
SlotAssignmentType	Обеспечивает связь слота с поставленным оборудованием.	struct, показывающая номер слота и тип подклю-чаемого элемента	
SoftwareDownload TrackingObjectIdType	Идентифицирует выполняющийся или неудачно завершившийся процесс загрузки программ.	TrackObjectI dType	
SONETSDHLinePMHistory Data	Задает структуру записи, содержащей данные о ходе процесса, собранные за 15-минутный интервал в точке контроля rsTTPF.	struct	
SONETSDHSection AdaptationPM HistoryData	Задает структуру записи, содержащей данные о ходе процесса, собранные за 15-минутный интервал в точках контроля au3CTPF или au4CTPF.	struct	
SONETSDHSectionPathPM HistoryData	Задает структуру записи, содержащей данные о ходе процесса, собранные за 15-минутный интервал в точках контроля msTTPF, vc3CTPF или vc4CTPF.		

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
SpanningTreeIndType	Этот булевский элемент данных указывает, включен ли алгоритм разветвляющегося дерева. TRUE означает, что включен.	boolean	
SpecificProblems	Этот элемент данных задает листинг кодовых значений для всех указанных проблем, связанных с условиями неудачного завершения или сигналами тревоги.	Последовательность long	Значения определяются поставщиком.
SSADTIndType	Этот булевский элемент данных указывает, выбран ли механизм гарантированной передачи данных; значение TRUE говорит о том, что выбран.	boolean	
SSCSParameterProfile1 Type	Этот элемент данных задает значения для профиля типа SSCS "профиль параметров 1".	struct	
SSCSParameterProfile2 Type	Этот элемент данных задает значения для профиля типа SSCS "профиль параметров 2".	struct	
SSCSTypeType	Этот элемент данных идентифицирует тип SSCS для AAL.	enum	
SSTEDIndType	Этот булевский элемент данных указывает, выбраны ли механизмы обнаружения ошибок передачи; значение TRUE говорит о том, что выбраны.	boolean	
StateChangeDefinition Type	Этот элемент данных задает листинг изменений переменных состояния и статуса, вызываемых условиями отказа, вызвавшими сигнал тревоги.	AttributeChangeSetType	Взят из Рекомендации МСЭ-Т X.780.
StatusAttributeType	Пара, показывающая тип значения состояния (StatusValueType) и процент завершения резервного копирования или восстановления конфигурационных данных, связанных с NE.	struct	
StatusValueType	Задает выбор значений между ProceduralStatusSetType и OtherStatus, указывающих аспекты завершения.	union	
StructuredDataTransfer Type	Этот булевский элемент данных указывает, сконфигурирован ли режим структурированной передачи данных (SDT) на AAL.	boolean	
SubType	Этот элемент данных идентифицирует подтип AAL.	enum	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
SupplyPowerIndType	Этот элемент данных указывает, подается ли напряжение на связанный порт.	boolean	
SWPValueType	Указывает точку контроля и связанные с ней параметры скользящего окна.	struct	
SynchChangeIndType	Этот булевский элемент данных указывает, передается ли через соединение синхронизация изменений в операции SSCS; значение TRUE говорит о том, что передается.	boolean	
SystemTimingType	Определяет первичный и вторичный источники времени для NE.	struct из struct	
T303Type	Этот элемент данных определяет максимальную длину интервала времени в миллисекундах, в течение которого ONT будет ждать ответа на сообщение SETUP , найденное на уровне 3 TMC или CSC.	enum	
T396Type	Этот элемент данных определяет максимальную длину интервала времени, в течение которого ONT будет ждать ответа на сообщение SETUP после истечения первоначального таймера T303.	enum	
TargetType	Этот элемент данных содержит листинг сетевых ресурсов.	Последовательность struct из ManagedEntityIdType и string	Позволяет делать выбор на уровне системы, NE, типа подключаемых элементов или указанного слота.
TargetActivityType	Обеспечивает связь уровней activityLevel, activityType и административного домена с пользователем или группой пользователей.	struct	
TCAdaptionProtocol MonitoringPM HistoryData	Задает структуру записи, перечисляющей данные о функционировании, собранные в течение 15-минутного интервала в точках контроля tcAdaptorF.		
TestHistoryType	Этот элемент данных показывает архивную информацию о завершенном процессе тестирования в системе управления поставщика.	struct	
TestIterationNumType	Этот элемент данных задает количество повторов теста.	short	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
TestTrackingObjectId Type	Этот элемент данных определяет выполняемый или незавершенный тест в системе управления поставщика.	TrackingObjectId	
ThresholdDataProfile Type	Этот элемент данных представляет собой листинг пар: копии thresholdDataProfile и типа точки контроля.	Последовательность struct	
ThresholdDataType	Этот элемент данных перечисляет имена параметров функционирования с пороговыми значениями для каждого из них.	struct	
ThresholdInfoType	Этот элемент данных предоставляет информацию о событии тревоги, связанном с качеством обслуживания, и обеспечивает идентификацию параметра функционирования, его наблюдаемого значения и наибольшего и наименьшего значений, что определяет диапазон пороговых значений.	struct	
ThresholdsList	Этот элемент данных перечисляет отношения между типами конечных точек контроля и связанным с ними профилем пороговых данных.	Последовательность struct Первый компонент показывает тип точки контроля, а второй содержит ссылку на threshold Data profile	
ThresholdsType	Этот элемент данных задает связь типов точек контроля с именем их профиля пороговых данных.	struct	
TimerCULengthType	Этот элемент данных задает значение для "комбинированного использования" таймера Timer_CU.	long long	
TimingReferenceType	Этот элемент данных определяет, как получается интервал таймера.	enum	
TotalEgressBandwidth Type	Этот элемент данных определяет общую величину диапазона (полосы частот) точки выхода для интерфейса ATM.	long long	
TotalIngressBandwidth Type	Этот элемент данных определяет общую величину диапазоны точки входа для интерфейса ATM.	long long	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
TrackingObjectIdType	Этот элемент данных помогает определить состояние запрошенного процесса, время завершения которого составляет более нескольких секунд.	unsigned long	Примеры включают TransferTracking ObjectIdType, SoftwareDown loadTracking ObjectType и TestTracking ObjectIdType.
TrafficDescriptor ProfileType	Этот элемент данных задает значения для профиля типа "дескриптор трафика" (Traffic Descriptor).	struct	
TransferTrackingObject IdType	Идентифицирует незавершенный процесс передачи файла в системе управления поставщика.	string	
TriggerTimeMatrixType	Определяет время срабатываний, связанное с периодом расписания.	struct	
UNIProfileType	Этот элемент данных определяет значения для профиля типа UNI.	struct	
UPCNPCDisagreementPM HistoryDataType	Этот элемент данных задает элементы для данного типа записей.	struct	
UPCNPCIndicatorType	Этот булевский элемент данных определяет, выполняется ли политика для всех соединений в интерфейсе.	boolean	
UsageStateType	Этот элемент данных нумерует значения для использования в переменной состояния.	enum	Допустимые значения: бездействие, активен, занят. Определены в Рекоменда-ции МСЭ-Т X.780.
UserGroupId	Это задаваемое оператором имя для группирования пользователей в системе управления поставщика.	UserLabel Type	Не может быть пустой строкой.
UserGroupType	Задает метку для группы пользователей, перечисляет пользователей – членов группы и нумерует их процессы.	struct из UserGroupId Type, UserId SeqType и Target Activity SeqType	
UserIdType	UserLabel, присваиваемая пользователям системы управления поставщика.	UserLabel	
UserLabelType	Задает идентификатор, созданный и поддерживаемый оператором или OMS для связи с ресурсом, управляемым системой управления поставщика.	string	
UserLoginPolicyType	Определяет политику управления входом пользователя.	struct	

Таблица A.1/Q.834.4 – Элементы данных и определения

Имя элемента данных	Определение	Синтаксис	Комментарий
UserLoginPolicy ViolationReasonType	Определяет одну причину сброса присвоения пароля пользователю.	enum	
UserType	Задает userId и группу, к которой принадлежит пользователь, и присвоенные пользователю виды деятельности.	struct из UserIdType, UserGroupId SeqType, и Target ActivitySeq Type	
VersionType	Задает один идентификатор версии оборудования или программного обеспечения для системы посредством объекта managedEntityIdType.	struct	
VoicePMHistoryDataType	Этот элемент данных определяет элементы записи данного типа.	struct	
VoiceServicesProfile AAL1Type	Этот элемент данных задает значения для профиля типа "речевые службы" (Voice Services) с использованием AAL1.	struct	
VoiceServicesProfile AAL2ProfileType	Этот элемент данных задает значения для профиля типа "речевые службы" (Voice Services) с использованием AAL2.	struct	
VPVCPMHistoryDataType	Этот элемент данных определяет элементы записи данного типа.	struct	
ПРИМЕЧАНИЕ. – Если модуль Q834Build предполагается использовать при управлении другими типами технологий доступа, то либо должен быть расширен список пронумерованных значений, либо может быть изменен в сторону сокращения синтаксис с константами, идентифицирующими типы конструируемых сетевых элементов.			

Приложение B

Исключительные ситуации

В таблице B.1 приведен список всех возможных исключительных ситуаций и обстоятельств, при которых они могут возникнуть в результате выполнения одной или нескольких операций данной спецификации интерфейса.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
AccessDenied	Система не дает доступа к данному объекту интерфейса.
ActivationCompleted	Указывает, что активация программного обеспечения выполнена, так что процесс активации не может быть отменен.
ActivationFailure	Процесс активации программного обеспечения завершился неудачно.
ActivityCompleted	Активация программного обеспечения выполнена и не может быть отменена.
ActivityInProgress	Эта исключительная ситуация возникает, если работа программного обеспечения инициирована и не может быть отменена.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
AddressLabelMismatch	Установленный NE не имеет текущего адреса DCN, указанного в запросе.
APONLayerFailure	Имело место аварийное завершение APON протокола выделения диапазонов между OLT и конструируемым промежуточным узлом.
BackupInProgress	Эта исключительная ситуация возникает, если запрос выдан во время выполнения резервного копирования.
CannotAssignManagedEntityId	Система управления поставщика не смогла установить ManagedEntityId для OLT, указывая тем самым, что не может управлять данным OLT.
CannotRetrieveUserLabel	Система управления поставщика не смогла считать метку пользователя, заданную OLT.
CollectionLimitation	Система управления поставщика не может собрать данные за заданный интервал времени из-за ограничений реализации.
CollectionPeriodPast	Если время окончания периода меньше или равно текущему.
CommFailure	Имел место отказ канала DCN между NE и системой управления поставщика.
ConnectionAlreadyExists	В подсети уже существует соединение с теми же конечными точками.
ConnectionCountExceeded	Максимальное количество соединений для OLT или порта PON было превышено данным запросом на предоставление услуги.
DCNTimeout	Канал связи DCN между по крайней мере одним NE и системой управления поставщика настолько перегружен, что текущее состояние или информация о состоянии не могут быть переданы за установленное системой время синхронизации.
DeniedAccess	Система не дает доступа к NE.
DuplicateProfileName	Эта исключительная ситуация возникает, если новое имя профиля дублирует уже существующее.
DuplicateSerialNumber	Существует другое оборудование того же типа с этим серийным номером.
DuplicateUserGroupId	Id уже используется другой группой пользователей.
DuplicateUserId	Профиль управления доступом уже установлен для данного идентификатора пользователя.
DuplicateUserLabel	Метка пользователя, указанная в запросе, используется для именования другого управляемого объекта в то же самое время.
EquipmentFailure	На NE в данный момент имеет место состояние отказа, мешающее завершению запрошенной транзакции.
HWServicesMismatch	Замененный NE не может поддерживать заданные службы.
InstallationFailure	Процесс установки программного обеспечения завершился аварийно.
InsufficientBW	Алгоритм CAC показывает, что запрошенная служба требует слишком большого диапазона для OLT.
InsufficientMemory	Недостаточно памяти на NE для загрузки программного модуля.
InsufficientPONBW	Для ONT или ONU не может быть выделен диапазон из-за недостатка диапазона в APONLink.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
InterfaceSpeedNotChangeable	Физический порт не может поддерживать новую скорость интерфейса, или скорость не может быть сконфигурирована.
IntervalCountTooLarge	Эта исключительная ситуация возникает, если запрошенные интервалы превышают максимум, поддерживаемый системой управления поставщика. Исключительная ситуация показывает максимальные допустимые интервалы контроля, поддерживаемые системой управления поставщика.
InvalidDCNAddress	Указан неверный адрес DCN.
InvalidEquipmentCode	Код оборудования не соответствует синтаксису.
InvalidExternalTime	Задано неверное внешнее время.
InvalidLocationId	Задан неверный LLID.
InvalidPort	Задан неверный порт PON.
InvalidProtectionScheme	Сетевой ресурс не поддерживает параметры защиты, указанные в контексте листинга портов, или если модули защиты представляют собой порты с несходными характеристиками физического пути.
InvalidScheduler	Значения параметров планировщика находятся за пределами определенного диапазона.
InvalidSerialNumSyntax	Синтаксис заданного серийного номера не соответствует правилам определения.
InvalidSlotAssignmentList	Ожидаемые правила определения слота нарушаются указанным присвоением.
InvalidSoftwareTracking Object	Программный объект, на который указана ссылка, не является самым последним по отношению к загрузке программного обеспечения на NE.
InvalidStartTime	Время запуска несовместимо с текущим временем, текущей матрицей времени срабатывания триггеров или с новым временем завершения.
InvalidStopTime	Новое время завершения несовместимо с текущей матрицей времени срабатывания триггеров, текущим временем или с новым временем запуска.
InvalidTestOperations	Запрошена неверная операция самотестирования.
InvalidTimeoutPeriod	Устанавливаемый период тайм-аута нарушает определение допустимых значений.
InvalidTrigger	Указанный триггер имеет значения, которые не могут быть интерпретированы планировщиком.
InvalidUserLabelSyntax	Указанная UserLabel не соответствует установленным правилам для userLabel.
LockedAlready	Административное состояние указанного управляемого объекта – "заблокирован", что не позволяет ему нормально выполнять свои функции.
MaxSubtendingNodesExceeded	Максимальное количество промежуточных узлов для указанного интерфейса PON превышено в результате запроса на резервирование или предоставление услуг.
NoResponse	Для ONT или ONU не может быть выделен диапазон, и отказ произошел из-за проблемы, отличной от неверного синтксиса серийного номера, протокола уровня APON либо повторной или неверной метки пользователя.
NoSuchRecords	Ни одна из записей в указанном наборе не соответствует критериям выборки.
NoSynchInProgress	Исключительная ситуация возникает, если нет выполняющегося процесса синхронизации.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
NotAvailableForTest	Указанный CTP недоступен для данного теста.
ParameterViolation	Эта исключительная ситуация возникает, если параметры конечной точки не соответствуют характеристикам протокола порта или если значения находятся вне диапазона либо дублируются.
ProfileInUse	Эта исключительная ситуация возникает, если профиль не может быть удален из-за того, что он все еще используется для задания характеристик управляемых объектов в пределах юрисдикции системы управления поставщика.
ProfileSuspended	Названный в запросе вызова профиль (профили) приостановлен для использования в системе управления поставщика оператором или OMS.
RecordSetExists	Набор записей, определенный параметрами запроса на создание, уже существует в системе управления поставщика.
RemainingContainedManaged Entities	Пакеты, содержащие схемы, или держатели оборудования еще не удалены.
RemainingReservations	Узел не может быть удален, поскольку еще существует соответствующее резервирование ресурса.
RemainingSubnetwork Connections	Узел подключаемого модуля не может быть удален, поскольку еще остаются соответствующие соединения в подсети.
ScheduleInUse	Еще есть процессы, планируемые указанным расписанием.
SlotAlreadyAssigned	Запрошенный слот уже поддерживается.
SoftwareLoadHardwareMismatch	Данные предыдущей конфигурации NE не могут быть загружены на NE, поскольку на NE имели место изменения в оборудовании, повлекшие невозможность выполнения этой операции.
SoftwareLoadHWMismatch	Указанные программы не могут быть загружены на оборудование, поскольку версия оборудования не соответствует загружаемым программам.
SoftwareNotYetInstalled	Программное обеспечение не может быть активировано, поскольку оно еще не установлено.
SoftwareTrackingObjectInUse	Имеются незавершенные программные процессы, связанные с данным объектом, и поэтому объект не может быть удален.
SourceUnreachable	Сервер, на котором находятся загружаемые программы, недоступен для OLT.
SynchNotScheduled	Эта исключительная ситуация возникает, если модификация расписания затрагивает NE, для которого предварительно не была выполнена операция синхронизации расписания.
Timeout	Длительность процесса достигла определенного в системе тайм-аута до того, как процесс завершился.
TooManyNEs	Система управления поставщика не может управлять еще одним OLT.
TooManyRecords	Количество записей, выбранных для ответа на запрос, превышает предварительно определенную величину.
UncontrolledTestInProgress	Самотестирование не может быть отменено, поскольку тест не является управляемым.
UnknownBackupProcess	Указанный объект передачи, идентифицирующий процесс передачи файла, неизвестен системе управления поставщика.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
UnknownConnection	Соединение в подсети неизвестно системе управления поставщика.
UnknownDestinationServer	Указанный конечный сервер недоступен для агента передачи.
UnknownHistoryDataType	Тип архивных данных неизвестен системе управления поставщика.
UnknownManagedEntity	Указанный управляемый объект неизвестен системе управления поставщика.
UnknownMonitoringPointTypes	MonitoringPointType неизвестен системе управления поставщика.
UnknownNE	Указанный NE неизвестен системе управления поставщика.
UnknownOption	Заданное значение параметра архива administrativeState – "в процессе отключения".
UnknownParameters	Заданный параметр контроля неизвестен системе управления поставщика.
UnknownPort	Указанный порт неизвестен системе управления поставщика.
UnknownProfiles	Эта исключительная ситуация возникает, если заданное имя профиля неизвестно системе управления поставщика и не может быть извлечено из хранилища профилей объектов.
UnknownRecordSet	Набор записей, указанный в запросе, неизвестен системе управления поставщика.
UnknownReservationId	Система управления поставщика не распознает данный Id резервирования.
UnknownRestoreProcess	Указанный объект передачи, идентифицирующий процесс передачи файла, неизвестен системе управления поставщика.
UnknownScheduler	Указанное расписание неизвестно системе управления поставщика.
UnknownService	Описываемая служба неизвестна системе управления поставщика.
UnknownServiceInstance	Экземпляр службы неизвестен системе управления поставщика.
UnknownSlot	Запрошенный слот неизвестен на NE.
UnknownSoftwareDownloadTrackObject	Указанный программный модуль (относящийся к программному обеспечению, поставляемому на NE) неизвестен системе управления поставщика.
UnknownSoftwareLoad	Указанный программный модуль нельзя найти.
UnknownSourceServer	Система управления поставщика и/или OLT не может установить связь с исходным сервером. Адрес DCN неизвестен или блокирован.
UnknownSystemTimingSource	Источник внешнего времени неизвестен системе управления поставщика.
UnknownTargets	Список процессов неизвестен системе управления поставщика.

Таблица B.1/Q.834.4 – Исключительные ситуации

Возникающая исключительная ситуация	Описание
UnknownTest	Тест, заданный с помощью идентификатора тестируемого объекта, неизвестен системе управления поставщика.
UnknownTransferProcess	Состояние указанного процесса передачи не может быть проверено из-за того, что неизвестно системе управления поставщика.
UnknownUserGroupId	Группа пользователей неизвестна системе управления поставщика.
UnknownUserIds	Эта исключительная ситуация возникает, если не удается распознать какой-либо userId.
UnrecognisedVersion	Заданная версия оборудования не соответствует известным значениям.
UserGroupNotEmpty	Непустая группа пользователей не может быть удалена.
UserLoginPolicyViolation	Заданное присвоение нового пароля пользователю нарушает политику входов пользователей, установленную в данный момент системой управления поставщика.

Приложение C

Файлы IDL

Если следующие файлы сохранены как текстовые, то любое их подмножество может быть успешно скомпилировано любым компилятором OMG IDL, соответствующим Спецификации CORBA 2.1 или более поздней спецификации при условии, что в это подмножество входят файл Q834Common.idl и стандартизованные файлы CosNaming.idl, CosNotifyFilter.idl, CosNotification.idl, X780.idl и CosNotifyComm.idl[11].

C.1 Q834AccessControl.idl

```
#ifndef __Q834_4_ACCESSCONTROL_DEFINED
#define __Q834_4_ACCESSCONTROL_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module AccessControl {
// Begin definitions from other idl files

// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrationDomainSeqType
                    AdministrationDomainSeqType;
```

[11] См. ссылки [18] и [19].

```
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;

#define AccessDenied Q834Common::AccessDenied

// End definitions from other idl files

// Local data types

     struct UserLoginPolicyType {
          short minUserId; // Minimum length of userid
          short minPassword; // Minimum length of password
          short passwordReuse;
          short loginAttempts;
          long passwordValidity;
          boolean alphanumeric;//?should password contain alphanumeric mixture
          boolean specialCharacters;
              //?should password contain special characters
          boolean repeatingCharacters; //?should password contain repeating
          boolean disallowUserId; //disallow username in password
     };

     struct SessionPolicyType {
          short sessionInactiveTime;
          short inactiveUserIdDisableTime;
          short multipleActiveLogins;
     };

     struct PasswordPolicyType {
          UserLoginPolicyType userLoginPolicy;
          SessionPolicyType sessionPolicy;
     };


     typedef sequence<UserIdType> UserIdSeqType;

     enum ActivityLevelType {
          monitorOnly,  // read
          allowedToExecute,  // write
          noAccess
     };

     typedef short ActivityType;



     struct TargetActivityType {
          ActivityType type;
          ActivityLevelType activityLevel;
          AdministrationDomainSeqType AdministrationDomainSeq;
     };

     typedef sequence<TargetActivityType> TargetActivitySeqType;

     enum UserLoginPolicyViolationReasonType {
          minUserId,
          minPassword,
          passwordReuse,
          loginAttempts,
          passwordValidity,
          alphanumeric,
          specialCharacters,
          repeatingCharacters,
```

```
        disallowUserId
    };


    typedef sequence<UserLoginPolicyViolationReasonType>
UserLoginPolicyViolationReasonSeqType;
        typedef sequence<UserLabelType> UserGroupIdSeqType;

    struct UserType {
        UserIdType userId;
        UserGroupIdSeqType userGroupIdSeq;
        TargetActivitySeqType TargetActivitySeq;
    };

    struct UserGroupType {
        UserLabelType userGroupId;
        UserIdSeqType userIdSeq;
        TargetActivitySeqType TargetActivitySeq;
    };

    typedef sequence<UserType> UserSeqType;
    typedef sequence<UserGroupType> UserGroupSeqType;


    // Local exceptions

    exception UnknownUserIds {
        UserIdSeqType userIdSeq;
    };
    exception DuplicateUserId {};
    exception UnknownUserGroupId {};
    exception DuplicateUserGroupId {};
    exception UnknownTargets {
        TargetActivitySeqType unknownTargetActivities;
    };
    exception UserGroupNotEmpty {};
    exception UserLoginPolicyViolation {
        UserLoginPolicyType userLoginPolicy;
        UserLoginPolicyViolationReasonSeqType reason;
    };
    // End local definitions

    valuetype AccessControlMgrValueType: itut_x780::ManagedObjectValueType {

        public PasswordPolicyType passwordPolicy; // GET
        public UserSeqType userList; // GET
        public UserGroupSeqType userGroupList; // GET
    };


    interface AccessControlMgr : itut_x780::ManagedObject {

    // define the activities
        const short ALL_ACTIVITIES = 0;
        const short ACCESS_CONTROL_MANAGEMENT = 1;
        const short ALARM_EVENT_CONFIGURATION_MANAGEMENT = 2;
        const short SCHEDULE_ACTIVITY = 3;
        const short SOFTWARE_DOWNLOAD = 4;
        const short TEST_CONTROL = 5;
        const short SYNCHRONISE_CURRENT_EVENT_LIST = 6;
        const short SYNCHRONISE_NE = 7;
        const short RANGE_NE = 8;
        const short REGISTER_SYSTEM = 9;
```

```
        const short RESERVE_RESOURCES = 10;
        const short PROFILE_MANAGEMENT = 11;
        const short PROVISION_NE = 12;
        const short PROVISION_TELEPHONY_SERVICE = 13;
        const short PROVISION_PACKETISED_DATA_SERVICES = 14;
        const short PROVISION_VIDEO_SERVICE = 15;
        const short PROVISION_LEASED_LINE_SERVICE = 16;
        const short BULK_TRANSFER = 17;
        const short HISTORY_DATA_COLLECTION = 18;
        const short CONTROL_ARCHIVING = 19;
        const short CONTROL_PERFORMANCE_MONITORING = 20;
        const short CONFIGURATION_BACKUP_RESTORE = 21;


    // См. в п. 9.1.1.1 описание поведения данной операции

        void setPasswordPolicy(
            in PasswordPolicyType passwordPolicy )
            raises ( AccessDenied);

    // См. в п. 9.1.1.2 описание поведения данной операции

        PasswordPolicyType passwordPolicyGet()
            raises (AccessDenied);

    // См. в п. 9.1.1.3 описание поведения данной операции

        UserSeqType userListGet ()
            raises (AccessDenied);

    // См. в п. 9.1.1.4 описание поведения данной операции

        UserGroupSeqType userGroupListGet ()
            raises (AccessDenied);

    // См. в п. 9.1.1.5 описание поведения данной операции

        UserType userGet (
            in UserIdType userId )
            raises (AccessDenied,
                UnknownUserIds);

    // См. в п. 9.1.1.6 описание поведения данной операции

        UserGroupType userGroupGet (
            in UserLabelType userGroupId)
            raises (AccessDenied,
                UnknownUserGroupId);


    // См. п. в 9.1.1.7 описание поведения данной операции

        void createUserGroup (
            in UserLabelType userGroupId,
            in TargetActivitySeqType targetAdditions)
            raises (DuplicateUserGroupId,
                UnknownTargets,
                AccessDenied);


    // См. в п. 9.1.1.8 описание поведения данной операции

        TargetActivitySeqType modifyUserGroup (
            in UserLabelType userGroupId,
```

```
          in TargetActivitySeqType targetAdditions,
          in TargetActivitySeqType targetDeletions)
          raises (UnknownUserGroupId,
               UnknownTargets,
               AccessDenied );


// См. в п. 9.1.1.9 описание поведения данной операции

     void deleteUserGroup (
          in UserLabelType userGroupId)
          raises (AccessDenied,
               UserGroupNotEmpty,
               UnknownUserGroupId );



// См. в п. 9.1.1.10 описание поведения данной операции

     void addUsersToGroup (
          in UserLabelType userGroupId,
          in UserIdSeqType userIdList )
          raises (AccessDenied,
               UnknownUserGroupId); // duplicate users are ignored


// См. в п. 9.1.1.11 описание поведения данной операции

     void deleteUsersFromGroup (
          in UserLabelType userGroupId,
          in UserIdSeqType userIdList )
          raises (AccessDenied,
          UnknownUserGroupId,
          UnknownUserIds);


// См. в п. 9.1.1.12 описание поведения данной операции

     TargetActivitySeqType getPermissionList (
          in UserIdType userId )
          raises (UnknownUserIds,
          AccessDenied) ;


// См. в п. 9.1.1.13 описание поведения данной операции

     TargetActivitySeqType modifyPermissionList (
          in UserIdType userId,
          in TargetActivitySeqType targetAdditions,
          in TargetActivitySeqType targetDeletions )
          raises (UnknownUserIds,
               UnknownTargets,
               AccessDenied);

// См. в п. 9.1.1.14 описание поведения данной операции

     void createUser (
          in UserIdType userId,
          in PasswordType password,
          in TargetActivitySeqType targetAdditions )
          raises (DuplicateUserId,
               UnknownTargets,
               AccessDenied,
               UserLoginPolicyViolation);
```

```
    // См. в п. 9.1.1.15 описание поведения данной операции

        void deleteUser (
            in UserIdType userId )
            raises (UnknownUserIds,
                AccessDenied);


    // См. в 9.1.1.16 описание поведения данной операции

        void resetPassword (
            in UserIdType userId,
            in PasswordType newPassword )
            raises (UnknownUserIds,
            UserLoginPolicyViolation,
            AccessDenied);

    }; // interface AccessControlMgr

}; // module AccessControl

}; // module q834_4

#endif
```

C.2 Q834Build.idl

```
#ifndef __Q834_4_BUILD_DEFINED
#define __Q834_4_BUILD_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module Build {

// begin definitions from other idl files

// From Q834Common
    typedef Q834Common::NameType NameType;
    typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
    typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
    typedef Q834Common::UserLabelType UserLabelType;

    typedef Q834Common::SlotAssignmentSeqType SlotAssignmentSeqType;
    typedef Q834Common::SerialNumType SerialNumType;
    typedef Q834Common::NameSeqType NameSeqType;
    typedef Q834Common::ExternalTimeType ExternalTimeType;
    typedef Q834Common::SystemTimingType SystemTimingType;
    typedef Q834Common::ReservationIdType ReservationIdType;
    typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
    typedef Q834Common::AdministrationDomainType AdministrationDomainType;
    typedef Q834Common::VersionType VersionType;
    typedef Q834Common::LoopbackLocationIdSeqType LoopbackLocationIdSeqType;
    typedef Q834Common::AdministrativeStateType AdministrativeStateType;
    typedef Q834Common::ServiceCategoryType ServiceCategoryType;
    typedef Q834Common::ConformanceDefType ConformanceDefType;
    typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
```

```
#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define UnknownProfiles Q834Common::UnknownProfiles
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define ParameterViolation Q834Common::ParameterViolation
#define ProfileSuspended Q834Common::ProfileSuspended
#define UnknownNE Q834Common::UnknownNE

// End definitions from other idl files

     // Local data types


     enum NEKindType {
          BPONOLT,
          BPONONT,
          BPONONU,
          BPONNT
     };




     struct ProtectionUnit
     {
          ManagedEntityIdType portId;
          boolean protectingInd; //TRUE = Protecting, FALSE = Protected
     };

     typedef sequence<ProtectionUnit> ProtectionUnitSeqType;

     enum ProtectionRatioType
     {
          oneForOne,
          oneForN, // N > 1
          mForN // M > 1
     };
               enum AllowedProtectionSwitchingType
     {
          automatic,
          manual,
          both
     };

        struct ProtectionParameterType
        {
          ProtectionRatioType protectionRatio;
          AllowedProtectionSwitchingType protectionSwitchingMethod;
          boolean revertiveInd; // TRUE = revertive
          long waitToRestore; // number of milliseconds before // reverting
        };

     enum SegmentEndpointIndType
     {
          segment,
          endpoint,
          none
     };
```

```
enum DirectionType
{
              unidirection,
              bidirection
};



struct OpticalWaveLengthArrayType {
              unsigned long wavelength;
              DirectionType direction;
     };
     typedef sequence <OpticalWaveLengthArrayType>
OpticalWaveLengthArraySeqType;


     // Local exceptions

     exception InvalidExternalTime {};
     exception UnrecognisedVersion {};
     exception DuplicateSerialNumber {};
     exception InvalidSlotAssignmentList {};
     exception RemainingContainedManagedEntities {
              ManagedEntityIdSeqType containedManagedEntities ;
     };
     exception UnknownNE {};
     exception UnknownSystemTimingSource {};
     exception RemainingReservations {
          ReservationIdSeqType remainingReservationList;
     };
     exception InvalidEquipmentCode {};
     exception SlotAlreadyAssigned {};
     exception UnknownSlot {};
     exception RemainingSubnetworkConnections {
          ManagedEntityIdSeqType containedManagedEntities ;
     };
          exception InterfaceSpeedNotChangeable {};
     exception InvalidProtectionScheme {};


     // End local definitions

     valuetype BuilderValueType: itut_x780::ManagedObjectValueType {

          public ManagedEntityIdSeqType createdNodes; // GET
     };

     interface Builder: itut_x780::ManagedObject {


     // См. в п. 9.2.1.1 описание поведения данной операции


ManagedEntityIdType buildNode (
              in NEKindType nEKind,
              in string supplierName,
              in string location,
              in VersionType hWVersion,
              in SerialNumType serialNum,
              in NameSeqType alarmSeverityProfiles,
              in NameSeqType thresholdDataProfiles,
              in SlotAssignmentSeqType slotAssignmentList,
              in ManagedEntityIdType port, // OLT PON port
              in string modelCode,
```

```
            in string systemTitle,
            in VersionSeqType softwareVersions,
            in UserLabelType nEUserLabel,
            in ExternalTimeType externalTime,
            in SystemTimingType systemTiming,
            in AdministrationDomainType administrationDomain )
            raises (UnrecognisedVersion,
                 InvalidSerialNumSyntax,
                 DuplicateSerialNumber,
                 UnknownProfiles,
                 UnknownManagedEntity,
                 DuplicateUserLabel,
                 AccessDenied,
                 InvalidExternalTime,
                 UnknownSystemTimingSource,
                 ProfileSuspended);

// См. п. в 9.2.1.2 описание поведения данной операции

     void assignUserLabelsToNE (
            in SerialNumType serialNum,
            in UserLabelType nEUserLabel,
            in AdministrationDomainType administrationDomain)
            raises (InvalidSerialNumSyntax,
                 DuplicateSerialNumber,
                 DuplicateUserLabel,
                 AccessDenied);

// См. в п. 9.2.1.3 описание поведения данной операции


void modifyNode (
            in ManagedEntityIdType managedEntityId,
            in SlotAssignmentSeqType newSlotAssignmentList,
            in NameSeqType newAlarmSeverityProfiles,
            in NameSeqType newThresholdDataProfiles,
            in ManagedEntityIdType port, // OLT PON Port
            in string newModelCode,
            in UserLabelType newNEuserLabel,
            in ExternalTimeType externalTime,
            in AdministrationDomainType administrationDomain )
            raises (UnknownManagedEntity,
                 UnknownNE,
                 InvalidSlotAssignmentList,
                 UnknownProfiles,
                 DuplicateUserLabel,
                 AccessDenied,
                 InvalidExternalTime,
                 ProfileSuspended);

// См. в п. 9.2.1.4 описание поведения данной операции

     void deleteNode (
            in ManagedEntityIdType managedEntityId )
            raises (UnknownNE,
                 RemainingContainedManagedEntities,
                 AccessDenied,
                 RemainingReservations,
                 RemainingSubnetworkConnections);



// См. в п. 9.2.1.5 описание поведения данной операции
```

```
     void modifyPort (
          in ManagedEntityIdType physicalPathTPId,
          in NameSeqType newAlarmSeverityProfiles,
          in NameSeqType newThresholdDataProfiles,
          in NameSeqType newPortProfiles,
          in string newFrameFormat,
          in AdministrativeStateType administrativeState,
          in OpticalWaveLengthArraySeqType newOpticalWavelengthArray,
          in LoopbackLocationIdSeqType newLoopbackLocationId,
          in unsigned long newInterfaceSpeed,
          in unsigned long aRCTimer)
          raises (UnknownManagedEntity,
               UnknownProfiles,
               AccessDenied,
               InterfaceSpeedNotChangeable,
               ProfileSuspended);

// См. в п. 9.2.1.6 описание поведения данной операции

     ManagedEntityIdType buildPlugInUnit (
          in ManagedEntityIdType nEId,
          in NameType alarmSeverityProfile,
          in UserLabelType plugInUnitUserLabel,
          in string modelCode,
          in ManagedEntityIdType equipmentHolder,
          in AdministrativeStateType administrativeState)
          raises (UnknownNE,
               DuplicateUserLabel,
               AccessDenied,
               UnknownManagedEntity,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               UnknownProfiles,
               ProfileSuspended);

// См. в п. 9.2.1.7 описание поведения данной операции

     ManagedEntityIdType modifyPlugInUnit (
          in ManagedEntityIdType plugInUnitId,
          in NameType newAlarmSeverityProfile,
          in string newModelCode,
          in ManagedEntityIdType newEquipmentHolder,
          in UserLabelType newPlugInUnitUserLabel,
          in AdministrativeStateType newAdministrativeState)
          raises (UnknownManagedEntity,
               UnknownProfiles,
               AccessDenied,
               InvalidEquipmentCode,
               SlotAlreadyAssigned,
               UnknownSlot,
               InvalidSlotAssignmentList,
               InvalidUserLabelSyntax,
               ProfileSuspended);



// См. в п. 9.2.1.8 описание поведения данной операции

     void deletePlugInUnit (
          in ManagedEntityIdType plugInUnitId )
          raises (UnknownManagedEntity,
               RemainingSubnetworkConnections,
```

```
                AccessDenied,
                RemainingReservations);

// См. в п. 9.2.1.9 описание поведения данной операции

     ManagedEntityIdType buildProtectionGrouping(
          in ProtectionParameterType protectionParameters,
          in ProtectionUnitSeqType protectionUnitList)
          raises (InvalidProtectionScheme,
               AccessDenied);

// См. в п. 9.2.1.10 описание поведения данной операции

     void modifyProtectionParameters(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionParameterType newProtectionParameters)
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// См. в п. 9.2.1.11 описание поведения данной операции

     void modifyProtectionUnitList(
          in ManagedEntityIdType protectionGroupingId,
          in ProtectionUnitSeqType deltaProtectionUnitList,
          in boolean addDeleteInd) // TRUE = add
          raises (UnknownManagedEntity,
               InvalidProtectionScheme,
                  AccessDenied);

// См. в п. 9.2.1.12 описание поведения данной операции
     void deleteProtectionGrouping(
          in ManagedEntityIdType protectionGroupingId)
          raises (UnknownManagedEntity,
                  AccessDenied);

// См. в п. 9.2.1.13 описание поведения данной операции
     ManagedEntityIdType buildBridge(
          in NameType mACBridgeProfile,
          in ManagedEntityIdType uplinkPort,
          in ManagedEntityIdSeqType uNIPortList)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// См. в п. 9.2.1.14 описание поведения данной операции
     void modifyBridgeProfile(
          in ManagedEntityIdType bridgeId,
          in NameType newMACBridgeProfile)
          raises (UnknownProfiles,
               AccessDenied,
               UnknownManagedEntity,
               ProfileSuspended);

// См. в п. 9.2.1.15 описание поведения данной операции
     void modifyBridgePortList(
          in ManagedEntityIdType bridgeId,
          in ManagedEntityIdSeqType deltaUNIPortList,
          in boolean addDeleteInd)
          raises (AccessDenied,
               RemainingSubnetworkConnections,
               UnknownManagedEntity);
```

```
     // См. в п. 9.2.1.16 описание поведения данной операции
          void deleteBridge(
               in ManagedEntityIdType bridgeId)
               raises (AccessDenied,
                    RemainingSubnetworkConnections,
                    UnknownManagedEntity);

     // См. в п. 9.2.1.17 описание поведения данной операции
          ManagedEntityIdType buildVPNetworkCTP(
               in ManagedEntityIdType port,
               in short vPI,
               in NameType trafficDescriptorProfileName,
               in ATMOverbookingFactorType overbookingFactor,
               in UserLabelType userLabel,
               in SegmentEndpointIndType segmentEndpointInd)
               raises (AccessDenied,
                    UnknownManagedEntity,
                    UnknownProfiles,
                    ParameterViolation,
                    ProfileSuspended);

     // См. в п. 9.2.1.18 описание поведения данной операции
          void deleteVPNetworkCTP (
               in ManagedEntityIdType vPNetworkCTP)
               raises (AccessDenied,
                    RemainingSubnetworkConnections,
                    UnknownManagedEntity);

     // См. в п. 9.2.1.19 описание поведения данной операции

          ManagedEntityIdSeqType createdNodesGet ()
               raises (AccessDenied);


     }; // interface Builder

}; // module Build

}; // module q834_4

#endif
```

C.3 Q834Common.idl

```
#ifndef __Q834_4_COMMON_DEFINED
#define __Q834_4_COMMON_DEFINED

#include "CosNaming.idl"
#include "CosNotifyFilter.idl"
#include "itut_x780.idl"
#include "TimeBase.idl"


#pragma prefix "itu.Int"

module q834_4 {

module Q834Common {

// Begin definitions from other idl files

// From CosNaming
     typedef CosNaming::Name NameType;
```

```
// From CosNotifyFilter
     typedef CosNotifyFilter::Filter FilterType;

// From TimeBase
     typedef TimeBase::UtcT UtcT;

// From X780

     typedef itut_x780::ProbableCauseType ProbableCauseType;
     typedef itut_x780::AdministrativeStateType AdministrativeStateType;
     typedef itut_x780::OperationalStateType OperationalStateType;
     typedef itut_x780::ProceduralStatusSetType ProceduralStatusSetType;
     typedef itut_x780::PerceivedSeverityType PerceivedSeverityType;
     typedef itut_x780::AvailabilityStatusSetType AvailabilityStatusSetType;
     typedef itut_x780::UsageStateType UsageStateType;
     typedef itut_x780::ControlStatusSetType ControlStatusSetType;
     typedef itut_x780::SpecificProblemSetType SpecificProblemSetType;
     typedef itut_x780::BackedUpStatusType BackedUpStatusType;
     typedef itut_x780::AttributeChangeSetType AttributeChangeSetType;
     typedef itut_x780::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types

     struct NamingComponentType {
          string type;  // managed entity type
          string id;
     };

     typedef sequence<NamingComponentType> RDNType;
     typedef sequence<RDNType> RDNSeqType;
     typedef sequence<NameType> NameSeqType;

     enum IdType {
          none,
          x780_fineGrained,
          x780_coarseGrained
     };

     typedef RDNType MEIdType;

     struct ManagedEntityIdType {
          IdType id;
          MEIdType mEId;
     };

     typedef sequence<ManagedEntityIdType> ManagedEntityIdSeqType;
     typedef string UserLabelType;
     typedef string SerialNumType;
     typedef string VersionType;
     typedef string PlugInUnitType;
     typedef sequence<UserLabelType> UserLabelSeqType;

     typedef UserLabelType AdministrationDomainType;
     typedef sequence<AdministrationDomainType> AdministrationDomainSeqType;

     typedef string DCNAddressType;

     typedef unsigned short RecordKindType;


     typedef   string    PlugInType;
```

```
// SlotAssignmentList
struct SlotAssignmentType {
     short number;
     PlugInType plugIn; // empty string implies that the slot is not
                    // provisioned for any particular type of plug-in unit
};

typedef sequence <SlotAssignmentType> SlotAssignmentSeqType;
typedef string ReservationIdType;
typedef sequence<ReservationIdType> ReservationIdSeqType;
typedef string ServiceInstanceIdType;
typedef sequence<ServiceInstanceIdType> ServiceInstanceIdSeqType;

typedef UtcT GeneralizedTimeType;
typedef GeneralizedTimeType ExternalTimeType;

enum SystemTimingSourceType {
     internalTimingSource, // freerun
     remoteTimingSource, // external
     slaveTimingTerminationSignal
};

typedef ManagedEntityIdType TimingSourceIdType ;

struct SystemTimingComponentType {
     SystemTimingSourceType type;
     TimingSourceIdType  id;
};

struct SystemTimingType {
     SystemTimingComponentType Primary;
     SystemTimingComponentType Secondary;
};


typedef UserLabelType UserIdType;
typedef string PasswordType;

/*
Все нули означают циклический запрос, направленный всем точкам соединения,
имеющим LLID в потоке. Все единицы означают циклический запрос, направленный
конечной точке (конечной точке сегмента или соединения).
'x6A'H означает отсутстве назначенных CP для цикла, и поэтому цикл не должен
выполняться. Все другие значения для LLID означают циклический запрос,
направленный в указанное LLID место.
*/
typedef   sequence<octet,16> LoopbackLocationIdSeqType;

typedef string FilenameType; // specifies the complete path

enum StatusType {
     procedural_status,
     other
};

enum OtherStatusType {
     completed_success,
     completed_failure,
     activityInProgress,
     unknownstatus
};
```

```
    union StatusValueType switch (StatusType) {
         case procedural_status: ProceduralStatusSetType proceduralStatusSet;
         case other: OtherStatusType otherStatus;
    };

    struct StatusAttributeType {
         StatusValueType valueOfStatus;
         ManagedEntityIdType ne;
         short percentComplete;
    };

    typedef sequence<StatusAttributeType> StatusAttributeSeqType;

    typedef  unsigned long TrackingObjectIdType;

    typedef TrackingObjectIdType TransferTrackingObjectIdType;

    typedef any RecordType; // based on recordType (Values defined in
Q834Common::RecordSetType) Need definition of recordType and individual type of
records
    typedef sequence<RecordType> RecordSeqType;


    typedef unsigned long long NotificationIdentifierType; // will be replaced
                        // by NotifIDType defined in X.780 when that definition
is changed from "long" to "long long"

    typedef sequence<NotificationIdentifierType> NotificationIdentifierSeqType;


    typedef stringMonitoredParameterType; // Values defined in
Q834Common::MonitoringParameter
    typedef unsigned short MonitoringKindType; // values defined in
Q834Common::PMCategory interface

    struct EndPointType {
         ManagedEntityIdType portId;
         any endPointParameters;
         NameSeqType serviceCharacteristicsProfiles;
    };

    /*
    endPointParameters: зависящие от службы параметры, структуры, задаваемые
    как часть реализации. Соединение ATM, например, может иметь параметры VPI,
    VCI.
    */


    enum AlarmStatusComponentType {
         AS_UnderRepair,
         AS_Critical,
         AS_Major,
         AS_Minor,
         AS_AlarmOutstanding
    };

    typedef sequence<AlarmStatusComponentType> AlarmStatusSeqType;

    struct EquipmentHolderAddressType {
         short shelfNumber;
         short slotNumber;
    };

    /*
```

```
     Если оборудование расположено на полке, номер слота равен 0. Если
     оборудованием владеет слот, тогда номер слота и номера полок больше или
     равны 1.
     */

     /*
     Поставщик обеспечивает настройку интерфейса DiscoveryEventSupplier
     Путем  определения имен постоянной строки plugInUnit. Оператор должен
     обеспечить последовательность и недублирование при нескольких решениях
     поставщика.
     */

     typedef sequence<PlugInUnitType> PlugInUnitSeqType;



struct NEFSANType {
          ManagedEntityIdType managedEntityId;
          AdministrativeStateType administrativeState;
          OperationalStateType operationalState;
          GeneralizedTimeType externalTime;
          string locationName;
          string supplierName;
          VersionType hardwareVersion;
          VersionSeqType softwareVersions;
          string serialNumber;
          NameSeqType alarmSeverityAssignmentProfileNames;
          AlarmStatusSeqType alarmStatusValues;
          NameSeqType thresholdDataNames;
          ManagedEntityIdSeqType supportedByManagedEntityList;
          UserLabelType userLabel;
     };
     /*
     SupportedByManagedEntityList должен включать копию программы,
     управляющей NE.
     */

     /*
     Затем определяются структуры (struct), составляющие основу инвентарной
     информации оборудования, обнаруживаемой при установке и действиях по
     изменению размещения оборудования. Любая из этих структур впоследствии
     идентифицируется как Mestruct в спецификации интерфейса
     DiscoveryEventSupplier.
     */

     struct OLTType {
          NEFSANType nEFSAN;
          ManagedEntityIdSeqType subtendingNEFSANList;
     };

     struct ONTType {
          NEFSANType nEFSAN;
          ManagedEntityIdType upstreamNEFSAN;
     };

     struct ONUType {
          NEFSANType nEFSAN;
          ManagedEntityIdType upstreamNEFSAN;
          ManagedEntityIdSeqType subtendingNEFSANList;
     };
```

```
struct NTType {
     NEFSANType nEFSAN;
     ManagedEntityIdType upstreamNEFSAN;
};

struct EquipmentHolderFType {
     ManagedEntityIdType equipmentHolderFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType equipmentHolderAddress;
     boolean slotStatus;
     PlugInUnitSeqType expectedPlugInUnits;
     string SoftwareLoad;
     NameType alarmSeverityAssignmentProfileName;
     AlarmStatusSeqType alarmStatusValues;
     OperationalStateType operationalState;
};

struct PlugInUnitFType {
     ManagedEntityIdType plugInUnitFId;
     ManagedEntityIdType containingNEId;
     EquipmentHolderAddressType containingSlotAddress;
     AdministrativeStateType administrativeState;
     AvailabilityStatusSetType availabilityStatus;
     OperationalStateType operationalState;
     string modelCode;
     string functionCode;
     string supplierName;
     VersionType hardwareVersion;
     string serialNumber;
     short portCount;
     NameSeqType alarmSeverityAssignmentProfileNames;
     NameSeqType thresholdDataNames;
     UserLabelType circuitPackUserLabel;
     ManagedEntityIdSeqType supportedByManagedEntityList;
};

enum ServiceCategoryType {
     CBR,
     UBR,
     RTVBR,
     NRTVBR,
     AdaptiveBR,
     GFR
};

enum ConformanceDefType {
     CBR1,
     UBR1,
     UBR2,
     VBR1,
     VBR2,
     VBR3,
     ABR,
     GFR1,
     GFR2
};

struct ATMOverbookingFactorType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType conformanceDef;
     short overbookingFactor; // percentage: 100 = no overbooking
};

struct AlarmLogRecordType {
```

```
        long long recordId;
        ManagedEntityIdType mEId;
        GeneralizedTimeType loggingTime;
        short probableCause;
        PerceivedSeverityType severity;
        GeneralizedTimeType eventTime;
        ManagedEntityIdType backupEntityId;
        boolean backedupStatus;
        boolean serviceAffectingInd;
        ServiceInstanceIdSeqType affectedServices;
        NotificationIdentifierType notificationId;
        NotificationIdentifierSeqType correlatedNotifications;
        string monitoredParameter;
        long long thresholdValue;
        long long observedValue;
        string additionalText;
    };

    struct SecurityAlarmLogRecordType {
        long long recordId;
        ManagedEntityIdType mEId;
        GeneralizedTimeType loggingTime;
        short securityAlarmCause; // values defined in X.780
        GeneralizedTimeType eventTime;
        ManagedEntityIdType securityAlarmDetector;
        ManagedEntityIdType serviceUser;
        ManagedEntityIdType serviceProvider;
        NotificationIdentifierType notificationId;
        NotificationIdentifierSeqType correlatedNotifications;
        string additionalText;
    };

    struct ServiceOutageRecordType {
        long long recordId;
        ServiceInstanceIdType affectedService;
        GeneralizedTimeType loggingTime;
        GeneralizedTimeType outageStartTime;
        GeneralizedTimeType outageEndTime;
        NotificationIdentifierSeqType correlatedNotifications;
    };

    struct AAL1PMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long HeaderErrors;
        unsigned long long LostCells;
        unsigned long long CellMisinsertion;
        unsigned long long BufferUnderflows;
        unsigned long long SequenceViolations;
        unsigned long long SDTPtrReframes;
        unsigned long long SDTPtrParityCheckFailures;
    };

    struct AAL2PMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId; unsigned long long CPSInPkts;
        unsigned long long CPSOutPkts;
        unsigned long long BufferUnderflow;
```

```
     unsigned long long BufferOverflow;
     unsigned long long ParityErrors;
     unsigned long long SeqNumErrors;
     unsigned long long CPS_OSFMismatchErrors;
     unsigned long long CPS_OSFErrors;
     unsigned long long CPSHECErrors;
     unsigned long long OversizedSDUErrors;
     unsigned long long ReassemblyErrors;
     unsigned long long HECOverlapErrors;
     unsigned long long UUIErrors;
     unsigned long long CIDErrors;
};

struct AAL5PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long SumOfInvalidCSFieldErrors;
     unsigned long long CRCViolations;
     unsigned long long BufferOverflows;
     unsigned long long EncapProtocolErrors;
};

struct APONPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ES;
     unsigned long long FEES;
};

struct ATMTrafficLoadHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long CellsReceived;
     unsigned long long CellsTransmitted;
};

struct DS1PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ESP;
     unsigned long long BESP;
     unsigned long long SESP;
     unsigned long long UASP;
     unsigned long long ESPFE;
     unsigned long long BESPFE;
     unsigned long long SESPFE;
     unsigned long long UASPFE;
};

struct DS3PMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
```

```
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long ESL;
    unsigned long long SESL;
    unsigned long long CVCPorCVPP;
    unsigned long long ESCPPorESPP;
    unsigned long long SESCPPorSESPP;
    unsigned long long UASCPPorUASPP;
};

struct E1PMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long ESP;
    unsigned long long BESP;
    unsigned long long SESP;
    unsigned long long UASP;
    unsigned long long ESPFE;
    unsigned long long BESPFE;
    unsigned long long SESPFE;
    unsigned long long UASPFE;
};

struct EthernetHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long SingleCollisionFrame;
    unsigned long long MultipleCollisionFrames;
    unsigned long long SQE;
    unsigned long long DeferredTransmission;
    unsigned long long LateCollision;
    unsigned long long ExcessiveCollision;
    unsigned long long InternalMACTransmitError;
    unsigned long long CarrierSenseError;
    unsigned long long BufferOverflows;
    unsigned long long AlignmentError;
    unsigned long long FrameTooLongs;
    unsigned long long FCSErrors;
    unsigned long long InternalMACReceiveError;
};

struct MACBridgePMHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
    unsigned long long NumberofSuppressedIntervals;
    unsigned long long BridgeLearningEntryDiscard;
};

struct MACBridgePMPortHistoryDataType {
    long long recordId;
    ManagedEntityIdType monitoringPoint;
    GeneralizedTimeType periodEndTime;
    boolean suspectIntervalFlag;
    NameType ThresholdDataId;
```

```
     unsigned long long ForwardedFrame;
     unsigned long long DelayExceededDiscard;
     unsigned long long MTUExceededDiscard;
     unsigned long long ReceivedFrame;
     unsigned long long ReceivedAndDiscarded;
};

struct UpcNpcDisagreementPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long DiscardedCells;
     unsigned long long DiscardedCLP_0Cells;
     unsigned long long TaggedCLP_0Cells;
};

struct VoicePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long IncomingCallAttempts;
     unsigned long long OutgoingCallAttempts;
     unsigned long long VoicePortBufferOverflows;
     unsigned long long VoicePortBufferUnderflows;
};

struct VpVcPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long Lost0plus1UserInformationCells;
     unsigned long long Lost0UserInformationCells;
     unsigned long long MisinsertedUserInformationCells;
     unsigned long long Transmitted0plus1UserInformationCells;
     unsigned long long Transmitted0UserInformation;
     unsigned long long ImpairedBlock;
};

struct SONETSDHLinePMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
     unsigned long long SeverelyErroredSecondsP;
     unsigned long long BackgroundBlockErrorP;
     unsigned long long OutOfFrameSecondsP;
     unsigned long long UnavailableSecondsP;
};

struct SONETSDHSectionPathPMHistoryDataType {
     long long recordId;
     ManagedEntityIdType monitoringPoint;
     GeneralizedTimeType periodEndTime;
     boolean suspectIntervalFlag;
     NameType ThresholdDataId;
     unsigned long long ErroredSecondsP;
```

```
        unsigned long long SeverelyErroredSecondsP;
        unsigned long long BackgroundBlockErrorP;
        unsigned long long OutOfFrameSecondsP;
        unsigned long long UnavailableSecondsP;
        unsigned long long FailureCountP;
        unsigned long long ErroredSecondsTypeAP;
        unsigned long long ErroredSecondsTypeBP;
        unsigned long long ErroredSecondsPFE;
        unsigned long long SeverelyErroredSecondsPFE;
        unsigned long long BackgroundBlockErrorPFE;
        unsigned long long OutOfFrameSecondsPFE;
        unsigned long long UnavailableSecondsPFE;
        unsigned long long FailureCountPFE;
        unsigned long long ErroredSecondsTypeAPFE;
        unsigned long long ErroredSecondsTypeBPFE;
    };

    struct SONETSDHSectionAdaptationPMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long PointerJustificationHighCountP;
        unsigned long long PointerJustificationLowCountP;
    };

    struct TCAdaptationProtocolMonitoringPMHistoryDataType {
        long long recordId;
        ManagedEntityIdType monitoringPoint;
        GeneralizedTimeType periodEndTime;
        boolean suspectIntervalFlag;
        NameType ThresholdDataId;
        unsigned long long DiscardedCellsHECViolationP;
        unsigned long long ErroredCellsHECViolationP;
    };

//Common exceptions

    exception AccessDenied {};
    exception CommFailure {};
    exception ConnectionCountExceeded {};
    exception DuplicateUserLabel {};
    exception EquipmentFailure {};
    exception InsufficientBW {};
    exception InsufficientPONBW {
        ManagedEntityIdType ponPORT;
    };
    exception InvalidSerialNumSyntax {};
    exception InvalidUserLabelSyntax {};
    exception MaxSubtendingNodesExceeded {};
    exception Timeout {};
    exception UnknownManagedEntity {
        ManagedEntityIdType managedEntityId;
    };
    exception UnknownNE {
        ManagedEntityIdSeqType unknownNEs;
    };
    exception UnknownPort {};
    exception UnknownProfiles {};
    exception UnknownReservationId {};
    exception UnknownScheduler {};
    exception InvalidScheduler {};
    exception DCNTimeout {};
```

```
    exception UnknownDestinationServer {};
    exception UnknownServiceInstance {};
    exception DeniedAccess {};
    exception BackupInProgress {};
    exception InvalidStartTime {};
    exception InvalidStopTime {};
    exception ParameterViolation {};
    exception UnknownRecordSet {};
    exception SynchInProgress {};
    exception ProfileSuspended {
        NameSeqType profilesSuspended;
    };


    module ProbableCauseConst {

        const string moduleName = "q834_4::Q834Common::ProbableCauseConst";

        interface ProbableCause {
        /*
        См. X.780 probableCause в itut_x780::ProbableCauseConst.
        */

        // Additional ProbableCause values needed for BPON

             const unsigned short LOSS_OF_PATH_POINTER = 1;
             const unsigned short STS_PAYLOAD_LABEL_MISMATCH = 2;
             const unsigned short STS_PATH_UNEQUIPPED = 3;
             const unsigned short ALARM_INDICATION_SIGNAL = 4;
             const unsigned short REMOTE_FAILURE_INDICATION = 5;
             const unsigned short REMOTE_ALARM_INDICATION = 6;
             const unsigned short ALARM_INDICATION_SIGNAL_PATH = 7;
             const unsigned short
ALARM_INDICATION_SIGNAL_CUSTOMER_INSTALLATION = 8;
             const unsigned short LOSS_OF_SIGNAL_TO_ALL_ONU_ONU = 9;
             const unsigned short LOSS_OF_SIGNAL_ONU_OR_ONT = 10;
             const unsigned short LOSS_OF_ACKNOWLEDGEMENT_ONU_OR_ONT = 11;
             const unsigned short PLOAMCL_ONU_OR_ONT = 12; //Physical Layer OAM
Cell Loss
             const unsigned short LOCD_ONU_OR_ONT = 13; // Loss of Cell
Delineation
             const unsigned short CPHE_ONU_OR_ONT = 14; // Cell Phase Error
             const unsigned short PEE_ONU_OR_ONT = 15;
             const unsigned short RF_ONU_OR_ONT = 16; // Ranging Failure
             const unsigned short BED_ONU_OR_ONT = 17; // Block Error Detection
             const unsigned short SD_ONU_OR_ONT = 18; // Signal Degraded
             const unsigned short REI_ONU_OR_ONT = 19; // Remote Error
Indication
             const unsigned short UM_ONU_OR_ONT = 20; // Unknown Message
             const unsigned short LM_ONU_OR_ONT = 21; // Link Mismatch
             const unsigned short REMOTE_DEFECT_INDICATION = 22; // Signal
Degraded
             const unsigned short MAJOR_POWER_FAILURE = 23;
             const unsigned short
REMOTE_ALARM_INDICATION_FAR_END_CUSTOMER_INSTALLATION = 24;
             const unsigned short LOSS_OF_ATM_CELL_DELINEATION = 25;
             const unsigned short LOW_BATTERY_THRESHOLD = 26;
             const unsigned short DIAGNOSTIC_TEST_FAILURE = 27; // См. в below
comment
             const unsigned short LOSS_OF_DCN_LINK = 28;
             const unsigned short CELL_STARVATION = 29;
             const unsigned short UNEXPECTED_PLUGIN = 30;
             const unsigned short IMPROPER_CARD_REMOVAL = 31;
             const unsigned short SLOT_CARD_MISMATCH = 32;
```

```
            const unsigned short LOS_LAN = 33; // Loss of carrier on Bridge
LAN port
            const unsigned short PERSISTENT_IMPAIRMENT = 34;
        }; // interface ProbableCause
    }; // module ProbableCauseConst

    interface MonitoringParameter {
        /*
        Названия параметров контролируемых характеристик и трафика
        */

        const string allParameters = "AllParameters";
        const string aAL1HeaderErrors = "AAL1HeaderErrors";
        const string allTypesCellsDiscarded = "AllTypesCellsDiscarded";
        const string aTMProtocolErrors = "ATMProtocolErrors";
        const string bESFEP = "BESFEP";
        const string bESP = "BESP";
        const string bufferOverflows = "BufferOverflows";
        const string bufferUnderflows = "BufferUnderflows";
        const string cellDelineationAnomalies = "CellDelineationAnomalies";
        const string cellMisinsertion = "CellMisinsertion";
        const string cRCViolations = "CRCViolations";
        const string cVCP = "CVCP";
        const string cVL = "CVL";
        const string cVPP = "CVPP";
        const string cVS = "CVS";
        const string discardedAllTypeCellsduetoNPC =
"DiscardedAllTypeCellsduetoNPC";
        const string discardedAllTypeCellsduetoUPC =
"DiscardedAllTypeCellsduetoUPC";
        const string discardedPriorityCellsduetoNPC =
"DiscardedPriorityCellsduetoNPC";
        const string discardedPriorityCellsduetoUPC =
"DiscardedPriorityCellsduetoUPC";
        const string encapProtocolErrors = "EncapProtocolErrors";
        const string eSCPP = "ESCPP";
        const string eSPONT = "ESPONT";
        const string eSL = "ESL";
        const string eSP = "ESP";
        const string eSPP = "ESPP";
        const string eSS = "ESS";
        const string excessiveCollisions = "ExcessiveCollisions";
        const string fCSErrors = "FCSErrors";
        const string frameTooLongs = "FrameTooLongs";
        const string hECViolations = "HECViolations";
        const string impairedBlocks = "ImpairedBlocks";
        const string lateCollisions = "LateCollisions";
        const string lostCells = "LostCells";
        const string lostPriorityUserInformationCells =
"LostPriorityUserInformationCells";
        const string lostUserInformationCells = "LostUserInformationCells";
        const string misinsertedUserInformationCells =
"MisinsertedUserInformationCells";
        const string priorityCellsDiscarded = "PriorityCellsDiscarded";
        const string sDTPointerReframes = "SDTPointerReframes";
        const string sDTPointerParityCheckFailures =
"SDTPointerParityCheckFailures";
        const string sequenceViolations = "SequenceViolations";
        const string sESCPP = "SESCPP";
        const string sESPONT = "SESPONT";
        const string sESL = "SESL";
        const string sESP = "SESP";
        const string sESPP = "SESPP";
        const string sESS = "SESS";
```

```
     const string sumOfInvalidCSFieldErrors = "SumOfInvalidCSFieldErrors";
     const string uASPONT = "UASPONT";
     const string uASCPP = "UASCPP";
     const string uASP = "UASP";
     const string uASPP = "UASPP";
     const string incomingCallAttempts = "IncomingCallAttempts";
     const string outgoingCallAttempts = "OutgoingCallAttempts";
     const string voicePortBufferOverflows = "VoicePortBufferOverflows";
     const string voicePortBufferUnderflows = "VoicePortBufferUnderflows";
     const string cPSInPkts = "CPSInPkts";
     const string cPSOutPkts = "CPSOutPkts";
     const string bufferUnderflow = "BufferUnderflow";
     const string bufferOverflow = "BufferOverflow";
     const string parityErrors = "ParityErrors";
     const string seqNumErrors = "SeqNumErrors";
     const string cPS_OSFMismatchErrors = "CPS_OSFMismatchErrors";
     const string cPS_OSFErrors = "CPS_OSFErrors";
     const string cPSHECErrors = "CPSHECErrors";
     const string oversizedSDUErrors = "OversizedSDUErrors";
     const string reassemblyErrors = "ReassemblyErrors";
     const string hECOverlapErrors = "HECOverlapErrors";
     const string uUIErrors = "UUIErrors";
     const string cIDErrors = "CIDErrors";

}; // interface MonitoringParameter

interface PMCategory {
     const unsigned short DS1_PM = 1;
     const unsigned short DS3_PM = 2;
     const unsigned short E1_PM = 3;
     const unsigned short E3_PM = 4;
     const unsigned short VP_PM = 5;
     const unsigned short VC_PM = 6;
     const unsigned short ETHERNET_PM = 7;
     const unsigned short TC_ADAPTOR_PM = 8;
     const unsigned short VOICE_PM = 9;
     const unsigned short APON_PM = 10;
     const unsigned short MACBRIDGE_PM = 11;
     const unsigned short MACBRIDGEPORT_PM = 12;
     const unsigned short ATMTRAFFICLOAD_PM = 13;
     const unsigned short AAL1_PM = 14;
     const unsigned short AAL2_PM = 15;
     const unsigned short AAL5_PM = 16;
     const unsigned short UPCNPC_PM = 17;
     const unsigned short DBA_PM = 18;
     const unsigned short SONET_SDH_LINE_PM = 19;
     const unsigned short SONET_SDH_SECTION_PATH_PM = 20;
     const unsigned short SONET_SDH_SECTION_ADAPTATION_PM = 21;
     const unsigned short CALL_STATS_PM = 22;
     const unsigned short VIDEO_PM = 23;

}; // interface PMCategory

interface RecordSetType {

// Beginning of Values for RecordKindType
// Values 1-99 reserved for HistoryDataType

     const unsigned short DS1PMHISTORYDATA = 1;
     const unsigned short DS3PMHISTORYDATA = 2;
     const unsigned short E1PMHISTORYDATA = 3;
     const unsigned short E3PMHISTORYDATA = 4;
     const unsigned short VPVCPMHISTORYDATA = 5;
     const unsigned short AAL1PMHISTORYDATA = 6;
```

```
          const unsigned short AAL2PMHISTORYDATA = 7;
          const unsigned short AAL5PMHISTORYDATA = 8;
          const unsigned short UPCNPCDISAGREEMENTPMHISTORYDATA = 9;
          const unsigned short ETHERNETPMHISTORYDATA = 10;
          const unsigned short VOICEPMHISTORYDATA = 11;
          const unsigned short MACBRIDGEPORTPMHISTORYDATA = 12;
          const unsigned short MACBRIDGEPMHISTORYDATA = 13;
          const unsigned short APONPMHISTORYDATA = 14;
          const unsigned short SONETSDHLINEPMHISTORYDATA = 15;
          const unsigned short SONETSDHSECTIONADAPTATIONPMHISTORYDATA = 16;
          const unsigned short SONETSDHSECTIONPATHPMHISTORYDATA = 17;
          const unsigned short TCADAPTATIONPROTOCOLMONITORINGPMHISTORYDATA = 18;
          const unsigned short ALARMLOGRECORD = 100;
          const unsigned short SECURITYALARMLOGRECORD = 101;
          const unsigned short SERVICEOUTAGERECORD = 102;

     // End of Values for RecordKindType

     }; // interface RecordSetType

interface PhysicalLayerLoopback {

     // Beginning of Values for LoopbackTestType

          const unsigned short LINELOOPBACK = 1;
          const unsigned short PAYLOADLOOPBACK = 2;
          const unsigned short INWARDLOOPBACK = 3;
          const unsigned short DUALLOOPBACK = 4;
          const unsigned short FACILITYLOOPBACK = 5;
          const unsigned short TERMINALLOOPBACK = 6;

     // End of Values for LoopbackTestType

     }; // interface  PhysicalLayerLoopback

};   // module Q834Common

}; // module q834_4
#endif
```

C.4 Q834ControlArchive.idl

```
#ifndef __Q834_4_CONTROLARCHIVE_DEFINED
#define __Q834_4_CONTROLARCHIVE_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ControlArchive {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::FilterType FilterType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::RecordType RecordType;
```

```
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;
     typedef Q834Common::RecordKindType RecordKindType;


#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define Timeout Q834Common::Timeout
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types

     enum FullActionType {
          halt, // indicates that the log should stop writing any more records
if the log is full
          wrap // indicates that the log should delete old records if the log is
full.
     };

     typedef unsigned long long MaxSizeType;
     typedef unsigned short SizeThresholdType; // 0-100%
     typedef unsigned long long CurrentSizeType;

     struct RecordSetStatusType {
          CurrentSizeType currentSize;
          OperationalStateType operationalState;
          MaxSizeType maxSize;
          SizeThresholdType sizeThreshold;
          NameType filterName;
          FullActionType fullAction;
          AdministrativeStateType administrativeState;
          RecordKindType recordKind;
          UserLabelType recordSetUserLabel;
     };


     enum CreationModeType {
          operatorDefined,
          initialList,
          either
     };

// Local exceptions

     exception RecordSetExists {ManagedEntityIdType recordSetId;};
     exception LockedAlready {};
     exception UnknownOption {};
     exception NoSuchRecords {};
     exception TooManyRecords {};

// End local definitions



     interface RecordSetMgr : itut_x780::ManagedObject {


     // См. в п. 9.4.1.1 описание поведения данной операции

          ManagedEntityIdType createLog (
               in UserLabelType recordSetUserLabel,
               in AdministrativeStateType administrativeState,
```

```
          in NameType filterName,
          in FullActionType fullAction,
          in MaxSizeType maxSize,
          in SizeThresholdType sizeThreshold)
          raises (RecordSetExists,
               DuplicateUserLabel,
               AccessDenied);

// См. в п. 9.4.1.2 описание поведения данной операции

     ManagedEntityIdType createArchive (
          in UserLabelType recordSetUserLabel,
          in AdministrativeStateType administrativeState,
          in RecordKindType recordKind,
          in MaxSizeType maxSize)
          raises (RecordSetExists,
               DuplicateUserLabel,
               AccessDenied);

// См. в п. 9.4.1.3 описание поведения данной операции


     RecordSetStatusType getStatusAttributes (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);

// См. в п. 9.4.1.4 описание поведения данной операции


     void suspendArchive (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);

// См. в п. 9.4.1.5 описание поведения данной операции

     void resumeArchive (
          in ManagedEntityIdType recordSetId)
          raises (AccessDenied, UnknownRecordSet);

// См. в п. 9.4.1.6 описание поведения данной операции

     void deleteArchive (
          in ManagedEntityIdType recordSetId )
          raises (UnknownRecordSet,
               AccessDenied );

// См. в п. 9.4.1.7 описание поведения данной операции

     void purgeArchive (
          in ManagedEntityIdType recordSetId )
          raises (UnknownRecordSet,
               AccessDenied );


// См. в п. 9.4.1.8 описание поведения данной операции

     RecordSeqType selectRecords (
          in FilterType SelectionFilter,
          in ManagedEntityIdType recordSetId)
          raises (UnknownRecordSet,
               Timeout,
               NoSuchRecords,
               AccessDenied,
               TooManyRecords);
```

```
     // См. в п. 9.4.1.9 описание поведения данной операции

          ManagedEntityIdSeqType recordSetListGet (
               in CreationModeType creationMode)
               raises (AccessDenied);


     // См. в п. 9.4.1.10 описание поведения данной операции

          void changeUserLabel(
               in ManagedEntityIdType recordSetId,
               in UserLabelType newUserLabel)
               raises (UnknownRecordSet,
                    AccessDenied,
                    DuplicateUserLabel);

     }; // interface RecordSetMgr

}; // module ControlArchive

}; // module q834_4

#endif
```

C.5 Q834SoftwareDownload.idl

```
#ifndef __Q834_4_SOFTWAREDOWNLOAD_DEFINED
#define __Q834_4_SOFTWAREDOWNLOAD_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SoftwareDownload {

// begin definitions from other idl files

// From Q834Common
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::PlugInUnitType PlugInUnitType;
     typedef Q834Common::FilenameType  FilenameType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownNE Q834Common::UnknownNE
#define DeniedAccess Q834Common::DeniedAccess
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define Timeout Q834Common::Timeout
```

```
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStartTime Q834Common::InvalidStartTime

// End definitions from other idl files

// Local data types

     typedef TrackingObjectIdType SoftwareDownloadTrackingObjectIdType;

     typedef sequence<SoftwareDownloadTrackingObjectIdType>
SoftwareDownloadTrackingObjectIdSeqType;
     typedef sequence<FilenameType> FilenameSeqType;

     struct VersionsType {
          ManagedEntityIdType resourceId;
          VersionType softwarePrimary;
          VersionType softwareStandBy;
          VersionType hardware;
     };

     typedef sequence<VersionsType> VersionsSeqType;

     struct TargetType {
          ManagedEntityIdType containingSystem;
          ManagedEntityIdType containingNE; // empty sequence means everything
          string plugInUnitType; // empty string here implies to any suitable
plugInUnitType
          ManagedEntityIdType slot; // empty sequence here means any slot.
     }; //string is supplied by the supplier with Release Notes.

     struct DownloadStatusType
     {
             ManagedEntityIdType targetId;
             StatusValueType deliveryStatus;
             StatusValueType commitStatus;
             StatusValueType activationStatus;
             StatusValueType revertStatus;
     };

        typedef sequence<DownloadStatusType> DownloadStatusSeqType;

// Local exceptions


     exception InvalidExternalTime {};
     exception UnrecognisedTarget {};
     exception InsufficientMemory {};
     exception SoftwareLoadHWMismatch {};
     exception SourceUnreachable {};
     exception UnknownSoftwareLoad {};
     exception InstallationFailure {};
     exception UnknownSoftwareDownloadTrackingObject {};
     exception SoftwareNotYetInstalled {};
     exception ActivationFailure {};
     exception ActivationCompleted {};
     exception ActivityCompleted {};
     exception ActivityInProgress {};
     exception InvalidSoftwareTrackingObject {};
     exception SoftwareTrackingObjectInUse {};

// End local definitions

     valuetype DownloadMgrValueType: itut_x780::ManagedObjectValueType {
```

```
          public SoftwareDownloadTrackingObjectIdSeqType
ScheduledSoftwareDownloadTrackingObjectList; // GET
          public SoftwareDownloadTrackingObjectIdSeqType
OnDemandSoftwareDownloadTrackingObjectList; // GET

     };

     interface DownloadMgr : itut_x780::ManagedObject {

     // См. в п. 9.5.1.1 описание поведения данной операции

          SoftwareDownloadTrackingObjectIdType deliverDistSWGlobal (
               in FilenameSeqType softwareSet,
               in DCNAddressType softwareSourceAddr,
               in UserIdType userId,
               in PasswordType password,
               in ManagedEntityIdSeqType deliverDistTargets)
               raises (CommFailure,
                    UnrecognisedTarget,
                    InsufficientMemory,
                    SoftwareLoadHWMismatch,
                    SourceUnreachable,
                    UnknownSoftwareLoad,
                    Timeout,
                    AccessDenied,
                    DeniedAccess);


     // См. в п. 9.5.1.2 описание поведения данной операции

          SoftwareDownloadTrackingObjectIdType deliverDistSWSpecific (
               in FilenameSeqType softwareSet,
               in DCNAddressType softwareSourceAddr,
               in UserIdType userId,
               in PasswordType password,
               in TargetType deliverDistTarget)
               raises (CommFailure,
                    UnrecognisedTarget,
                    InsufficientMemory,
                    SoftwareLoadHWMismatch,
                    SourceUnreachable,
                    UnknownSoftwareLoad,
                    Timeout,
                    AccessDenied,
                    DeniedAccess);

     // См. в п. 9.5.1.3 описание поведения данной операции

          void deleteSoftwareDownloadTrackingObject (
               in SoftwareDownloadTrackingObjectIdType id)
               raises (UnknownSoftwareDownloadTrackingObject, AccessDenied);

     // См. в п. 9.5.1.4 описание поведения данной операции

          void commit (
               in SoftwareDownloadTrackingObjectIdType id,
               in TargetType commitTarget)
               raises (InstallationFailure,
                    UnknownSoftwareDownloadTrackingObject,
                    AccessDenied,
                    UnrecognisedTarget);

     // См. в п. 9.5.1.5 описание поведения данной операции
```

```
     void activate (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType activateTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget);

// См. в п. 9.5.1.6 описание поведения данной операции

     void revert (
          in SoftwareDownloadTrackingObjectIdType id,
          in TargetType revertTarget)
          raises (UnknownSoftwareDownloadTrackingObject,
               SoftwareNotYetInstalled,
               ActivationFailure,
               AccessDenied,
               UnrecognisedTarget,
               InvalidSoftwareTrackingObject);

// См. в п. 9.5.1.7 описание поведения данной операции

     DownloadStatusSeqType getStatus (
          in SoftwareDownloadTrackingObjectIdType id)
          raises (UnknownSoftwareDownloadTrackingObject,
               AccessDenied);

// См. в п. 9.5.1.8 описание поведения данной операции

     SoftwareDownloadTrackingObjectIdType scheduleDeliverDist (
          in FilenameSeqType softwareSet,
          in DCNAddressType softwareSourceAddr,
          in UserIdType userId,
          in PasswordType password,
          in ManagedEntityIdSeqType deliverDistTargets,
          in GeneralizedTimeType deliverDistStartTime)
          raises (SoftwareLoadHWMismatch,
               UnknownScheduler,
               AccessDenied,
               InvalidStartTime);

// См. в п. 9.5.1.9 описание поведения данной операции

     void scheduleCommit (
          in SoftwareDownloadTrackingObjectIdType
          deliverDistSoftwareDownloadTrackingObjectId,
          in GeneralizedTimeType commitStartTime)
          raises (UnknownSoftwareDownloadTrackingObject,
               UnknownScheduler,
               SoftwareNotYetInstalled,
               AccessDenied,
               InvalidStartTime);

// См. в п. 9.5.1.10 описание поведения данной операции

     void scheduleActivate (
          in SoftwareDownloadTrackingObjectIdType id,
          in UserLabelType activateSchedulerName,
          in GeneralizedTimeType activateStartTime)
          raises (UnknownSoftwareDownloadTrackingObject,
               InvalidStartTime,
               SoftwareNotYetInstalled,
               AccessDenied,
```

```
                    InvalidScheduler );

     // См. в п. 9.5.1.11 описание поведения данной операции

          void cancelScheduledSoftwareActivity (
               in SoftwareDownloadTrackingObjectIdType id)
               raises (UnknownSoftwareDownloadTrackingObject,
                    ActivityCompleted,
                    ActivityInProgress,
                    AccessDenied);

     // См. в п. 9.5.1.12 описание поведения данной операции

          SoftwareDownloadTrackingObjectIdSeqType
     scheduledSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);

     // См. в п. 9.5.1.13 описание поведения данной операции

          SoftwareDownloadTrackingObjectIdSeqType
     onDemandSoftwareDownloadTrackingObjectList ()
               raises (AccessDenied);


     }; // interface DownloadMgr



     interface VersionRepository : itut_x780::ManagedObject {

     // См. в п. 9.5.2.1 описание поведения данной операции

          VersionsSeqType retrieveVersions (
               in ManagedEntityIdType containingManagedEntityId)
               raises (CommFailure,
                    UnknownManagedEntity,
                    AccessDenied);

     // См. в п. 9.5.2.2 описание поведения данной операции

          boolean validateNEVersion (
               in ManagedEntityIdType managedEntityId,
               in VersionType proposedSoftware)
               raises (UnknownNE, AccessDenied);

     // См. в п. 9.5.2.3 описание поведения данной операции

          boolean validatePlugInVersion (
               in ManagedEntityIdType plugInUnitId,
               in VersionType proposedSoftware)
               raises (UnknownManagedEntity, AccessDenied);

}; // interface VersionRepository

}; // module SoftwareDownload

}; // module q834_4

#endif
```

C.6 Q834EventPublisher.idl

```
#ifndef __Q834_4_EVENTPUBLISHING_DEFINED
#define __Q834_4_EVENTPUBLISHING_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module q834_4
{

module EventPublisher
{

// begin definitions from other idl files - filterable data value types


// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::VersionType VersionType;
     typedef Q834Common::ProceduralStatusSetType ProceduralStatusSetType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::OperationalStateType OperationalStateType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::ProbableCauseType ProbableCauseType; // Values defined
in Q834Common::ProbableCauseConst
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::NotificationIdentifierSeqType
NotificationIdentifierSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::EquipmentHolderFType EquipmentHolderFType;
     typedef Q834Common::PlugInUnitFType PlugInUnitFType;
     typedef Q834Common::UsageStateType UsageStateType;
     typedef Q834Common::ControlStatusSetType ControlStatusSetType;
     typedef Q834Common::SpecificProblemSetType SpecificProblemSetType;
     typedef Q834Common::BackedUpStatusType BackedUpStatusType;
     typedef Q834Common::AttributeChangeSetType AttributeChangeSetType;
     typedef Q834Common::SecurityAlarmCauseType SecurityAlarmCauseType;

// End definitions from other idl files

// Local data types




     typedef NotificationIdentifierSeqType CorrelatedNotificationType;

     typedef AttributeChangeSetType StateChangeDefinitionType;


     struct ThresholdInfoType {
          MonitoredParameterType monitoredParameter;
          unsigned long long observedValue;
          unsigned long long thresholdValueLow;
          unsigned long long thresholdValueHigh;
     };
```

```
    /*
    Значения monitoredParameter приведены в Q834Common::MonitoringParameter.
    В настоящее время поддерживаются только счетчики для контроля характеристик
    или трафика в технологии BPON; как следствие, observedValue и
    thresholdValueHigh ограничиваются неотрицательными целочисленными
    значениями, а thresholdValueLow равно 0.
    */

    typedef   unsigned short EquipmentType;



    enum ServiceAffectingType {
         serviceAffecting,
         nonServiceAffecting,
         unableToDetermine
    };


//End local data type definitions

// Local exceptions

// End local exceptions


    interface AlarmEventSupplier : itut_x780::ManagedObject {

         /* Отображения фиксированного заголовка структурированного события:
         domain_type устанавливается на "telecommunications",
         type_name устанавливается на "Alarm", а
         event_name - на одну из следующих постоянных строк,
         приведенных ниже.
         */

         const string communicationAlarm = "CommunicationAlarm";
         const string equipmentAlarm = "EquipmentAlarm";
         const string environmentalAlarm = "EnvironmentalAlarm";
         const string processingErrorAlarm = "ProcessingErrorAlarm";
         const string qualityOfServiceAlarm = "QualityOfServiceAlarm";

         /*
         Имена фильтруемых данных для размещения в части
         фильтруемого события в структурированном событии перечисляются
         в указанном ниже порядке.
         */

         const string alarmEmittingMEId = "AlarmEmittingMEId";
         const string eventTime = "EventTime";
         const string probableCause = "ProbableCause";
         const string specificProblems = "SpecificProblems";
         const string perceivedSeverity = "PerceivedSeverity";
         const string backUpStatus = "BackUpStatus";
         const string backUpManagedEntityId = "BackUpManagedEntityId";
         const string thresholdInfo = "ThresholdInfo";
         const string notificationIdentifier = "NotificationIdentifier";
         const string correlatedNotifications = "CorrelatedNotifications";
         const string stateChangeDefinition = "StateChangeDefinition";
         const string monitoredAttributes = "MonitoredAttributes";
         const string additionalText = "AdditionalText";
         const string serviceAffectingInd = "ServiceAffectingInd";

         /*
         Оставшиеся значения фильтруемых данных.
```

```
        /* Имена переменных состояния и статуса для StateChangeDefinition
        */

        const string administrativeState = "AdministrativeState";
        const string operationalState = "OperationalState";
        const string usageState = "UsageState";
        const string availabilityStatus = "AvailabilityStatus";
        const string proceduralStatus = "ProceduralStatus";
        const string controlStatus = "ControlStatus";
        const string alarmStatus = "AlarmStatus";


        /*
        Ниже приводится отображение в фильтруемые данные в
        структурированном событии для сигнала тревоги для связи,
        сигнала тревоги для оборудования, сигнала тревоги из-за ошибки
        обработки, сигнала тревоги из-за ухудшения среды и
        сигнала обработки из-за ухудшения качества обслуживания.

        {
        {"AlarmEmittingMEId", any (ManagedEntityIdType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ProbableCause", any (ProbableCauseType)},
        {"SpecificProblems", any (SpecificProblemSetType)},
        {"PerceivedSeverity", any (PerceivedSeverityType)},
        {"ServiceAffectingInd", any (ServiceAffectingType)},
        {"BackUpStatus", any (BackedUpStatusType)},
        {"BackUpManagedEntityId", any (ManagedEntityIdType)},
        {"ThresholdInfo", any (ThresholdInfoType)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"StateChangeDefinition", any (StateChangeDefinitionType)},
        {"AdditionalText", any (string)}
        }

        */

};  // interface AlarmEventSupplier


interface SecurityEventSupplier : itut_x780::ManagedObject {

        /*  Отображения фиксированного заголовка структурированного события:
        domain_type устанавливается на "telecommunications",
        type_name устанавливается на "SecurityEvent", а
        event_name – на одну из следующих постоянных строк,
        приведенных ниже.
        */

        const string integrityViolation = "IntegrityViolation";
        const string operationalViolation = "OperationalViolation";
        const string physicalViolation = "PhysicalViolation";
        const string securityEventViolation = "SecurityEventViolation";
        const string timeDomainViolation = "TimeDomainViolation";



        const string eventEmittingMEId = "EventEmittingMEId";
        const string eventTime = "EventTime";
        const string securityAlarmCause = "SecurityAlarmCause";
        const string securityAlarmDetector = "SecurityAlarmDetector";
        const string serviceUser = "ServiceUser";
        const string serviceProvider = "ServiceProvider";
        const string securityEventNotificationIdentifier =
```

```
        "NotificationIdentifier";
    const string correlatedNotifications = "CorrelatedNotifications";
    const string additionalText = "AdditionalText";
    /*
    Ниже приводится отображение в фильтруемые данные в структурированном
    событии для нарушений целостности (Integrity Violations),
    эксплуатационных нарушений (Operational Violations), физических
    нарушений (Physical Violations),нарушений события безопасности
    (Security Event Violations) и нарушений временной области
    (Time Domain Violations).


    {
    {"EventEmittingMEId", any (ManagedEntityIdType)},
    {"EventTime", any (GeneralizedTimeType)},
    {"SecurityAlarmCause", any (SecurityAlarmCauseType)},
    {"SecurityAlarmDetector", any (ManagedEntityIdType)},
    {"ServiceUser", any (ManagedEntityIdType)},
    {"ServiceProvider", any (ManagedEntityIdType)},
    {"NotificationIdentifier", any (NotificationIdentifierType)},
    {"CorrelatedNotifications", any (CorrelatedNotificationType)},
    {"AdditionalText", any (string)}
    }

    */

}; // interface SecurityEventSupplier


interface DiscoveryEventSupplier : itut_x780::ManagedObject {

    /*  Отображение фиксированного заголовка структурированного события:
    domain_type устанавливается на "telecommunications", а
    type_name устанавливается на "DiscoveryEvent", п
    event_name устанавливается на одну из следующих постоянных строк,
    приведенных ниже. Через этот интерфейс объявляется об изменениях
    только установленного оборудования.
    */

    const string managedEntityCreation = "ManagedEntityCreation";
    const string managedEntityDeletion = "ManagedEntityDeletion";


    const string managedEntityType = "ManagedEntityType";
    const string managedEntityAttributeValues =
    "ManagedEntityAttributeValues";
    const string discoveryNotificationIdentifier =
    "NotificationIdentifier";
    const string correlatedNotifications = "CorrelatedNotifications";
    const string additionalText = "AdditionalText";"


    // The following items are equipment types that are discovered by the
    // Supplier Management System and automatically revealed to the OMS.
    // The data structure passed on the notification is defined in
    // Q834Common.idl and the mapping below refers to it as MEstruct. More
    // specifically, here is the list of data structures currently
    // supported: OLTType, ONUType,ONTType, NTType, EquipmentHolderType,
    and PlugInUnitFType.

    const unsigned short OLT_NE = 1;
    const unsigned short ONT_NE = 2;
    const unsigned short ONU_NE = 3;
    const unsigned short NT_NE = 4;
```

```
        const unsigned short EQUIPMENT_HOLDER_F = 5;
        const unsigned short PLUG_IN_UNIT_F = 6;


        /*
        Ниже приводится отображение в фильруемые данные в
        структурированном событии для события открытия, которое
        включает создание  управляемого объекта.



        {
        {"ManagedEntityType", any (EquipmentType)},
        {"EventTime", any (GeneralizedTimeType)},
        {"ManagedEntityAttributeValues", any (MEstruct)},
        {"NotificationIdentifier", any (NotificationIdentifierType)},
        {"CorrelatedNotifications", any (CorrelatedNotificationType)},
        {"AdditionalText", any (string)}
        }

        */

    }; // interface DiscoveryEventSupplier

}; // module EventPublisher

}; // module q834_4

#endif
```

C.7 Q834MIBTransfer.idl

```
#ifndef __Q834_4_MIBTRANSFER_DEFINED
#define __Q834_4_MIBTRANSFER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module MIBTransfer {
// Begin definitions from other idl files


// From Q834Common

    typedef Q834Common::UserLabelType UserLabelType;
    typedef Q834Common::DCNAddressType DCNAddressType;
    typedef Q834Common::FilenameType FilenameType;
    typedef Q834Common::StatusAttributeSeqType StatusAttributeSeqType;
    typedef Q834Common::TransferTrackingObjectIdType
    TransferTrackingObjectIdType;
    typedef Q834Common::UserIdType UserIdType;
    typedef Q834Common::PasswordType PasswordType;
    typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define FileExists Q834Common::FileExists
```

```
#define CannotCreateFile Q834Common::CannotCreateFile
#define DeniedAccess Q834Common::DeniedAccess

// End definitions from other idl files

// Local data types

     typedef sequence<TransferTrackingObjectIdType>
TransferTrackingObjectIdSeqType;

// Local exceptions

     exception UnknownBackupProcess {};
     exception UnknownSourceServer {};
     exception UnknownRestoreProcess {};
     exception SoftwareLoadHardwareMismatch {};


// End local definitions

     valuetype MIBMoverValueType: itut_x780::ManagedObjectValueType {

          public TransferTrackingObjectIdSeqType transferTrackingObjectIdList;
// GET

     };

     interface MIBMover : itut_x780::ManagedObject {

     // См. в п. 9.7.1.1 описание поведения данной операции

          TransferTrackingObjectIdType startBackup (
               in ManagedEntityIdType nEManagedEntityId, // OLT
               in DCNAddressType destinationServerAddr,
               in UserIdType userId,
               in PasswordType password,
               in FilenameType destinationFile,
               in boolean overwriteExistingFile)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownDestinationServer,
                    CommFailure,
                    EquipmentFailure,
                    DeniedAccess);

     // См. в п. 9.7.1.2 описание поведения данной операции

          StatusAttributeSeqType getBackupStatus (
               in TransferTrackingObjectIdType id)
               raises (AccessDenied,
                    UnknownBackupProcess);


     // См. в п. 9.7.1.3 описание поведения данной операции

          TransferTrackingObjectIdType scheduleBackup (
               in ManagedEntityIdType nEManagedEntityId, // OLT
               in UserIdType userId,
               in PasswordType password,
               in UserLabelType schedulerName,
               in DCNAddressType destinationServerAddr,
               in FilenameType destinationFile,
```

```
            in boolean overwriteExistingFile)
            raises (AccessDenied,
                 UnknownNE,
                 UnknownScheduler,
                 UnknownDestinationServer,
                 InvalidScheduler);

// См. в п. 9.7.1.4 описание поведения данной операции

          void modifyBackupSchedule(
            in TransferTrackingObjectIdType id,
            in UserLabelType newSchedulerName)
            raises (AccessDenied,
                 UnknownBackupProcess,
                 UnknownScheduler,
                 InvalidScheduler);

// См. в п. 9.7.1.5 описание поведения данной операции

     void cancelScheduledBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess);

// См. в п. 9.7.1.6 описание поведения данной операции

     void abortBackup (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownBackupProcess,
               CommFailure,
               EquipmentFailure);

// См. в п. 9.7.1.7 описание поведения данной операции

     TransferTrackingObjectIdType startRestore (
          in ManagedEntityIdType nEManagedEntityId, // OLT
          in DCNAddressType sourceServerAddr,
          in UserIdType userId,
          in PasswordType password,
          in FilenameType sourceFile)
          raises (AccessDenied,
               UnknownNE,
               UnknownSourceServer,
               CommFailure,
               EquipmentFailure,
               DeniedAccess,
               SoftwareLoadHardwareMismatch );

// См. в п. 9.7.1.8 описание поведения данной операции

     StatusAttributeSeqType getRestoreStatus (
          in TransferTrackingObjectIdType id)
          raises (AccessDenied,
               UnknownRestoreProcess);
```

```
     // См. в п. 9.7.1.9 описание поведения данной операции

          TransferTrackingObjectIdSeqType transferTrackingObjectIdListGet ()
               raises (AccessDenied);

     }; // interface MIBMover

}; // module MIBTransfer

}; // module q834_4

#endif
```

C.8 Q834PerformanceManager.idl

```
#ifndef __Q834_4_PERFORMANCEMANAGER_DEFINED
#define __Q834_4_PERFORMANCEMANAGER_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module PerformanceManager {

     // begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::RecordSeqType RecordSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;
     typedef Q834Common::RecordKindType RecordKindType;


     #define AccessDenied Q834Common::AccessDenied
     #define UnknownNE Q834Common::UnknownNE
     #define UnknownManagedEntity Q834Common::UnknownManagedEntity
     #define UnknownProfiles Q834Common::UnknownProfiles
     #define UnknownServiceInstance Q834Common::UnknownServiceInstance
     #define UnknownScheduler Q834Common::UnknownScheduler
     #define CommFailure Q834Common::CommFailure
     #define InvalidScheduler Q834Common::InvalidScheduler
     #define EquipmentFailure Q834Common::EquipmentFailure
     #define ProfileSuspended Q834Common::ProfileSuspended

     // End definitions from other idl files

     // Local data types


     struct  MonitoringPointKindAndThresholdsType {
          MonitoringKindType monitoringType;
          NameType  thresholdDataProfileName;
     };
     /*
```

```
     ПРИМЕЧАНИЕ. – Отношения между ThresholdData и конкретными типами точек
     контроля описаны в Рекомендациях МСЭ-Т Q.834.1 и Q.834.2.
     */

     typedef sequence<MonitoringPointKindAndThresholdsType> ThresholdsSeqType;

     struct MonitoringPointAndThresholdsType {
          ManagedEntityIdType monitoringPoint;
          NameType  thresholdDataProfileName;
     };

     typedef sequence<MonitoringPointAndThresholdsType>
MonitoringPointThresholdsSeqType;

     typedef sequence<MonitoredParameterType> MonitoredParameterSeqType;

     typedef sequence<RecordSeqType> RecordsSeqType; // Implicitly grouped by
record type.

     typedef RecordKindType  HistoryDataType; // Values defined in
Q834Common::RecordSetType

     typedef ManagedEntityIdSeqType MonitoringPointSeqType;

     struct SWPValueType {
          ManagedEntityIdType monitoringPoint;
          short totConsecutiveIntvls;
          short persistenceMinimum;
     };

     typedef sequence< SWPValueType > SWPValueSeqType;


     struct ParameterSettingType {
          MonitoredParameterType monitoredParameter;
          short totConsecutiveIntvls;
          short persistenceMinimum;
     };

     typedef sequence<ParameterSettingType> ParameterSettingSeqType;

     // Local exceptions

     exception UnknownParameters {
          MonitoredParameterSeqType monitoredParameterList;
     };
     exception IntervalCountTooLarge {};
     exception UnknownMonitoringPointTypes {
          MonitoringPointSeqType monitoringPointKindList;
     };
     exception InvalidAssociation {};
     exception CollectionPeriodPast {};
     exception CollectionLimitation {};
     exception UnknownHistoryDataType {};

     // End local definitions


     interface ImpairmentPersistence : itut_x780::ManagedObject {

          // См. в п. 9.8.1.1 описание поведения данной операции

          void setSlidingWindowParameters (
               in ManagedEntityIdType nEManagedEntityId,
```

```
     in MonitoredParameterSeqType monitoredParameterList,
     in short totConsecutiveIntvls,
     in short persistenceMinimum,
     in boolean sysScopeInd)
     raises (UnknownNE,
          UnknownParameters,
          IntervalCountTooLarge,
          AccessDenied,
          CommFailure);

// См. в п. 9.8.1.2 описание поведения данной операции

void setSpecificSlidingWindowParameters (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in MonitoredParameterSeqType monitoredParameterList,
     in short totConsecutiveIntvls,
     in short persistenceMinimum,
     in boolean allowGlobalOverwrite)
     raises (UnknownNE,
          UnknownParameters,
          IntervalCountTooLarge,
          AccessDenied,
          UnknownManagedEntity,
          CommFailure,
          EquipmentFailure);

// См. в п. 9.8.1.3 описание поведения данной операции

SWPValueSeqType getSpecificSlidingWindowParameters (
     in ManagedEntityIdType nEManagedEntityId,
     in MonitoredParameterType monitoredParameter)
     raises (UnknownNE,
          UnknownParameters,
          CommFailure);

// См. в п. 9.8.1.4 описание поведения данной операции

void setThreshold (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in NameType thresholdDataProfileName)
     raises (UnknownNE,
          AccessDenied,
          UnknownManagedEntity,
          UnknownProfiles,
          InvalidAssociation,
          CommFailure,
          ProfileSuspended);

// См. в п. 9.8.1.5 описание поведения данной операции

void setThresholds (
     in ManagedEntityIdType nEManagedEntityId,
     in boolean sysScopeInd,
     in ThresholdsSeqType thresholdsList)
     raises (UnknownNE,
          UnknownProfiles,
          AccessDenied,
          UnknownMonitoringPointTypes,
          InvalidAssociation,
          CommFailure,
          ProfileSuspended);
```

```
        // См. в п. 9.8.1.6 описание поведения данной операции

        MonitoringPointThresholdsSeqType getThresholdValues (
             in ManagedEntityIdType nEManagedEntityId,
             in MonitoringKindType monitoringType)
             raises (UnknownNE,
                  UnknownMonitoringPointTypes,
                  CommFailure);

        // См. в п. 9.8.1.7 описание поведения данной операции

        ThresholdsSeqType getSystemThresholdsSetting(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, UnknownManagedEntity);

        // См. в п. 9.8.1.8 описание поведения данной операции

        ParameterSettingSeqType getSystemSWSettings(
             in ManagedEntityIdType nEManagedEntityId)
             raises (AccessDenied, UnknownManagedEntity);

    }; // interface ImpairmentPersistence


    interface ReportController : itut_x780::ManagedObject {

        // См. в п. 9.8.2.1 описание поведения данной операции

        void addCustomerMonitoringReporting  (
             in ManagedEntityIdType nEManagedEntityId,
             in ServiceInstanceIdType serviceInstanceId,
             in ManagedEntityIdType monitoringPoint,
             in GeneralizedTimeType stopTime,
             in HistoryDataType historyData,
             in short granularityPeriod) //in minutes
             raises (UnknownServiceInstance,
                  AccessDenied,
                  UnknownNE,
                  UnknownManagedEntity,
                  CollectionPeriodPast,
                  CollectionLimitation,
                  InvalidAssociation,
                  UnknownHistoryDataType,
                  CommFailure);

        // См. в п. 9.8.2.2 описание поведения данной операции

        void removeCustomerMonitoringReporting  (
             in ServiceInstanceIdType serviceInstanceId)
             raises (UnknownServiceInstance,
                  AccessDenied,
                  CollectionPeriodPast,
                  CommFailure);

        // См. в п. 9.8.2.3 описание поведения данной операции

        RecordsSeqType selectByServiceInstance (
             in ServiceInstanceIdType serviceInstanceId,
             in GeneralizedTimeType intervalStartTime,
             in GeneralizedTimeType intervalEndTime)
             raises (UnknownServiceInstance,
                  AccessDenied );
```

```
// См. в п. 9.8.2.4 описание поведения данной операции

MonitoringPointSeqType displayActiveReporting (
     in ServiceInstanceIdType serviceInstanceId)
     raises (UnknownServiceInstance,
          AccessDenied);

// См. в п. 9.8.2.5 описание поведения данной операции

void addNewMonitoringReporting (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType stopTime,
     in HistoryDataType historyData,
     in short granularityPeriod) //in minutes
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionPeriodPast,
          CollectionLimitation,
          InvalidAssociation,
          UnknownHistoryDataType,
          CommFailure);

// См. в п. 9.8.2.6 описание поведения данной операции

RecordsSeqType selectByMonitoringPoint (
     in ManagedEntityIdType monitoringPoint,
     in GeneralizedTimeType intervalStartTime,
     in GeneralizedTimeType intervalEndTime)
     raises (UnknownManagedEntity,
          AccessDenied);


// См. в п. 9.8.2.7 описание поведения данной операции

void createReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
     in ServiceInstanceIdType serviceInstance,
     in short granularityPeriod, //in minutes
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionLimitation,
          UnknownScheduler,
          InvalidAssociation,
          UnknownHistoryDataType,
          InvalidScheduler);

// См. в п. 9.8.2.8 описание поведения данной операции

void modifyReportingSchedule (
     in ManagedEntityIdType nEManagedEntityId,
     in ManagedEntityIdType monitoringPoint,
     in HistoryDataType historyData,
     in UserLabelType newSchedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownManagedEntity,
          CollectionLimitation,
          UnknownScheduler,
```

```
                    InvalidAssociation,
                    UnknownHistoryDataType,
                    InvalidScheduler);

          // См. в п. 9.8.2.9 описание поведения данной операции

          void cancelReportingSchedule (
               in ManagedEntityIdType nEManagedEntityId,
               in ManagedEntityIdType monitoringPoint,
               in HistoryDataType historyData,
               in UserLabelType schedulerName)
               raises (AccessDenied,
                    UnknownNE,
                    UnknownManagedEntity,
                    UnknownScheduler,
                    InvalidAssociation,
                    UnknownHistoryDataType);

     }; // interface ReportController

}; // module PerformanceManager

}; // module q834_4

#endif
```

C.9 Q834ProfileManager.idl

```
#ifndef __Q834_4_PROFILEMANAGER_DEFINED
#define __Q834_4_PROFILEMANAGER_DEFINED
#include "Q834Common.idl"
#pragma prefix "itu.Int"


module  q834_4
{

module ProfileManager
{

     // begin definitions from other idl files - filterable data value types
     typedef Q834Common::NameType NameType;
     typedef Q834Common::NotificationIdentifierType NotificationIdentifierType;
     typedef Q834Common::PerceivedSeverityType PerceivedSeverityType;
     typedef Q834Common::ProbableCauseType ProbableCauseType;
     typedef Q834Common::MonitoredParameterType MonitoredParameterType;
     typedef Q834Common::ServiceCategoryType ServiceCategoryType;
     typedef Q834Common::ConformanceDefType ConformanceDefType;
     typedef Q834Common::ATMOverbookingFactorType ATMOverbookingFactorType;
     typedef Q834Common::MonitoringKindType MonitoringKindType;

#define UnknownProfiles Q834Common::UnknownProfiles
#define AccessDenied Q834Common::AccessDenied

     // End definitions from other idl files

     // Local data types

     typedef unsigned short ProfileKindType;
```

```
struct ProfileInfoType {
     ProfileKindType profileKind;
     any  attributeValuestruct;
};

/*
Далее, структуры struct (и последовательности struct) определены путем
присвоения значений атрибутов для вновь созданных объектов профиля. Любая
из этих структур впоследствии идентифицируется в спецификации интерфейса
Profile Event Consumer как ProfileStruct.
*/

enum SubType  {
     NULL,
     VoicebandBasedOn64kbps,
     SynchronousCircuitEmulation,
     AsynchronousCircuitEmulation,
     HighQualityAudio,
     Video
};

enum CBRRateType {
     br44736kbps,
     br1544kbps,
     br2048kbps,
     brE3kbps,
     br64kbps,
     br2x64kbps,
     br3x64kbps,
     br4x64kbps,
     br5x64kbps,
     br6x64kbps,
     br7x64kbps,
     br8x64kbps,
     br9x64kbps,
     br10x64kbps,
     br11x64kbps,
     br12x64kbps,
     br13x64kbps,
     br14x64kbps,
     br15x64kbps,
     br16x64kbps,
     br17x64kbps,
     br18x64kbps,
     br19x64kbps,
     br20x64kbps,
     br21x64kbps,
     br22x64kbps,
     br23x64kbps,
     br24x64kbps,
     br25x64kbps,
     br26x64kbps,
     br27x64kbps,
     br28x64kbps,
     br29x64kbps,
     br30x64kbps,
     br31x64kbps
};

enum ClockRecoveryType {
     PhysInterface,
     SRTS,
```

```
     AdaptiveClock,
     LocalOsc
};

enum ForwardErrorCorrectionType {
     NoFEC,
     FECforLossSensitiveSigTransport,
     FECforDelaySensSigTransport
};

typedef boolean StructuredDataTransferType; //TRUE = STD has been chosen
typedef long long PartiallyFilledCellsType;
typedef long long CellLossIntegrationPeriodType;

struct AAL1ProfileType {
     SubType aAL1subtype;
     CBRRateType cBRRate;
     ClockRecoveryType clockRecovery;
     ForwardErrorCorrectionType forwardErrorCorrection;
     StructuredDataTransferType structuredDataTransfer;
     PartiallyFilledCellsType partiallyFilledCells;
}; // ProfileStruct for profile type = 1


typedef stringDefaultSSCSParameterProfile1PtrType;

typedef stringDefaultSSCSParameterProfile2PtrType;

struct    AAL2ProfileType {
     DefaultSSCSParameterProfile1PtrType defaultSSCSParameterProfile1Ptr;
     DefaultSSCSParameterProfile2PtrType defaultSSCSParameterProfile2Ptr;
}; // ProfileStruct for profile type = 2


struct    MaxCPCSSDUSizeType {
     long long forwardDirectionCPCS_SDU;
     long long backwardsDirectionCPCS_SDU;
};

enum AALModeType {
     MessageAssured,
     MessageUnassured,
     StreamingAssured,
     StreamingUnassured
};

enum SSCSType {
     NoSSC,
     DataSSCSonSSCOPAssured,
     DataSSCSonSSCOPNotAssured,
     FrameRelaySSCS
};

struct AAL5ProfileType {
     // ManagedEntityIdType mEId; ??
     MaxCPCSSDUSizeType maxCPCSSDUSize;
     AALModeType aALMode;
     SSCSType sSCS;
}; // ProfileStruct for profile type = 4


enum AppIdType {
     LES_CAS_noELCP,
     LES_PSTN_noELCP,
```

```
        LES_PSTN_ELCP,
        LES_DSS1forBRI_noELCP,
        LES_DSS1forBRI_ELCP,
        LES_CASforPOTS_DSS1forBRI_noELCP,
        LES_PSTNforPOTS_DSS1forBRI_noELCP,
        LES_PSTNforPOTS_DSS1forBRI_ELCP,
        LES_otherCCS,
        UnspecifiedLES
    };

    typedef long long MaxNumChannelsType;
    typedef long long MinimumChanIdType;
    typedef long long MaximumChanIdType;
    typedef long long MaxCPS_SDULengthType;
    typedef long long TimerCULengthType;

    struct AAL2PVCProfileType {
        MaxNumChannelsType maxNumChannels;
        MinimumChanIdType minimumChanId;
        MaximumChanIdType maximumChanId;
        MaxCPS_SDULengthType maxCPS_SDULength;
        TimerCULengthType timerCULength;
    }; // ProfileStruct for profile type = 3

    struct AlarmSeverityAssignProfileType {
        string eventName; //As defined in AlarmEventSupplier
        ProbableCauseType probableCauseValue; //Ditto as above
        PerceivedSeverityType serviceAffectingSeverity;
        PerceivedSeverityType nonServiceAffectingSeverity;
    }; // ProfileStruct for profile type = 5

    typedef sequence<AlarmSeverityAssignProfileType>
AlarmSeverityAssignProfileSeqType;
    typedef long long LocalMaxNumVPCSupportedType;
    typedef long long LocalMaxNumVCCSupportedType;
    typedef short LocalMaxNumVPIBitsType;
    typedef short LocalMaxNumVCIBitsType;
    typedef long long TotalEgressBandwidthType;
    typedef long long TotalIngressBandwidthType;
    typedef boolean UPCNPCIndicatorType; //TRUE = policing is on

    struct ATMNetworkAccessProfileType {
        LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
        LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
        LocalMaxNumVPIBitsType localMaxNumVPIBits;
        LocalMaxNumVCIBitsType localMaxNumVCIBits;
        TotalEgressBandwidthType totalEgressBandwidth;
        TotalIngressBandwidthType totalIngressBandwidth;
        UPCNPCIndicatorType uPCNPCIndicator;
    }; // ProfileStruct for profile type = 6


    typedef short LANType;
    typedef short EncapsulationProtocolType;
    typedef short PIDType;

    struct BridgedLANServiceProfileValues {
        LANType lAN;
        EncapsulationProtocolType encapsulationProtocol;
        PIDType pID;
    }; // ProfileStruct for profile type = 7


    typedef long long BufferedCDVToleranceType;
```

```
enum CASType {
     basic,
     e1Cas,
     SfCas,
     ds1EsfCas,
     j2Cas
};

typedef long long CableLengthType;

struct CESServiceProfileValues {
     BufferedCDVToleranceType bufferedCDVTolerance;
     CASType cAS;
}; // ProfileStruct for profile type = 8


enum DS1FramingType {
     SuperFrame,
     ExtendedSuperFrame,
     Unframed
};

enum DS1EncodingType {
     AMI,
     B8ZS
};

enum LoopbackCodeType {
     SmartJack,
     SmartJack_ONTInband,
     COInband
};

typedef boolean SupplyPowerIndType;

struct DS1ProfileValues {
     DS1FramingType dS1Framing;
     DS1EncodingType dS1Encoding;
     LoopbackCodeType loopbackCode;
     SupplyPowerIndType supplyPowerInd;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 9


enum DS3ApplicationType {
     CBitParity,
     M23
};

enum DS3EncodingType {
     B3ZS,
     CCHAN
};

struct DS3ProfileType {
     DS3ApplicationType dS3Application;
     DS3EncodingType dS3Encoding;
     CableLengthType cableLength;
}; // ProfileStruct for profile type = 10

typedef boolean DuplexType; //TRUE = full, FALSE = half
typedef boolean AutoDetectionIndType;
```

```
enum DataRateType {
     TenBT,
     HundredBT,
     ThousandBT,
     otherrate
};

typedef long long MaxFrameSizeType; //Fixed for Ethernet

typedef boolean DTEDCEType; // TRUE = DTE setting; FALSE = DCE setting.

struct EthernetProfileValues {
     DuplexType duplex;
     AutoDetectionIndType autoDetectionInd;
     DataRateType dataRate;
     MaxFrameSizeType maxFrameSize;
     DTEDCEType dTEDCE;
}; // ProfileStruct for profile type = 11


enum T303Type {
     m700,
     m1200,
     m1700,
     m2200,
     m2700,
     m3200,
     m3700,
     m4200,
     m4700
};

enum T396Type {
     ms700,
     ms1700,
     ms2700,
     ms3700,
     ms4700,
     ms5700,
     ms6700,
     ms7700,
     ms8700,
     ms9700,
     ms10700,
     ms11700,
     ms12700,
     ms13700,
     ms14700
};

struct IDLCCallProcessingProfileType {
     T303Type t303;
     T396Type t396;
}; // ProfileStruct for profile type = 12

typedef boolean ELCPIndType;

enum POTSSignallingType {
     PSTN,
     ChAS,
     CCS,
     UNKNOWN
};
```

```
enum BRISignallingType {
     DSS1,
     OtherCCS
};

typedef long long MaxNumCIDsType;
typedef long long MaxPacketLengthType;
struct ChannelWithSSCSPtrType {
     long channelIndex;
     boolean ptr1orPtr2;
};

typedef sequence<ChannelWithSSCSPtrType> ChanAndSSCSParaPtrSeqType;

struct LESProfileType {
     ELCPIndType eLCPInd;
     POTSSignallingType pOTSSignalling;
     BRISignallingType bRISignalling;
     MaxNumCIDsType maxNumCIDs;
     MaxPacketLengthType maxPacketLength;
     ChanAndSSCSParaPtrSeqType chanAndSSCSParaPtrSeq;
}; // ProfileStruct for profile type = 13


typedef boolean SpanningTreeIndType;
typedef short BridgePriorityType;
typedef short MaxAgeType;
typedef short HelloTimeType;
typedef short ForwardDelayType;

struct MACBridgeServiceProfileType {
     SpanningTreeIndType spanningTreeInd;
     BridgePriorityType bridgePriority;
     MaxAgeType maxAge;
     HelloTimeType helloTime;
     ForwardDelayType forwardDelay;
}; // ProfileStruct for profile type = 14

typedef long long SegmentLengthType;
typedef short RASTimerType;
typedef long longMaxSSSARSDULengthType;
typedef boolean SSTEDIndType;
typedef boolean SSADTIndType;

struct SSCSParameterProfile1Type {
     SegmentLengthType segmentLength;
     RASTimerType rASTimer;
     MaxSSSARSDULengthType maxSSSARSDULength;
     SSTEDIndType sSTEDInd;
     SSADTIndType sSADTInd;
}; // ProfileStruct for profile type = 15

enum ServiceCatType {
     Audio,
     Multirate,
     UNKNCategory
};

enum EncSrcType {
     ITUT,
     ATMForum,
     Proprietary
};
```

```
typedef short EncProfileIndexType;
typedef boolean AudioServIndType;
typedef short PCMEncType;
typedef boolean CMDataIndType;
typedef short CMMultiplierNumType;
typedef boolean FMDataIndType;
typedef long long FMMaxFrameLenType;
typedef boolean CASIndType;
typedef boolean DTMFIndType;
typedef boolean MFR1IndType;
typedef boolean MFR2IndType;
typedef boolean RateControlIndType;
typedef boolean SynchChangeIndType;
typedef boolean FaxDemodIndType;

struct SSCSParameterProfile2Type {
     ServiceCatType serviceCat;
     EncSrcType encSrc;
     EncProfileIndexType encProfileIndex;
     AudioServIndType audioServInd;
     PCMEncType pCMEnc;
     CMDataIndType cMDataInd;
     CMMultiplierNumType cMMultiplierNum;
     FMDataIndType fMDataInd;
     FMMaxFrameLenType fMMaxFrameLen;
     CASIndType cASInd;
     DTMFIndType dTMFInd;
     MFR1IndType mFR1Ind;
     MFR2IndType mFR2Ind;
     RateControlIndType rateControlInd;
     SynchChangeIndType synchChangeInd;
     FaxDemodIndType faxDemodInd;
}; // ProfileStruct for profile type = 16


typedef long long PCRIngressType;
typedef long long PCREgressType;
typedef long long CDVTPCRIngressType;
typedef long long CDVTPCREgressType;
typedef long long CDVTSCRIngressType;
typedef long long CDVTSCREgressType;
typedef long long SCRIngressType;
typedef long long SCREgressType;
typedef long long MaxBSIngressType;
typedef long long MaxBSEgressType;
typedef long long MFSIngressType;
typedef long long MFSEgressType;

struct TrafficDescriptorProfileType {
     ServiceCategoryType serviceCategory;
     ConformanceDefType conformanceDef;
     PCRIngressTypepCRIngress;
     PCREgressType pCREgress;
     CDVTPCRIngressType cDVTPCRIngress;
     CDVTPCREgressType cDVTPCREgress;
     CDVTSCRIngressType cDVTSCRIngress;
     CDVTSCREgressType cDVTSCREgress;
     SCRIngressType sCRIngress;
     SCREgressType sCREgress;
     MaxBSIngressType maxBSIngress;
     MaxBSEgressType maxBSEgress;
     MFSIngressType mFSIngress;
     MFSEgressType mFSEgress;
}; // ProfileStruct for profile type = 17
```

```
    typedef string LoopbackLocCodeType;

    struct UNIProfileType {
         LocalMaxNumVPCSupportedType localMaxNumVPCSupported;
         LocalMaxNumVCCSupportedType localMaxNumVCCSupported;
         LocalMaxNumVPIBitsType localMaxNumVPIBits;
         LocalMaxNumVCIBitsType localMaxNumVCIBits;
         LoopbackLocCodeType loopbackLocCode;
    }; // ProfileStruct for profile type = 18


    enum AnnouncementType {
         Silence,
         ReorderTone,
         FastBusy,
         VoiceAnnouncement,
         OtherAnnouncementType,
         UNKNAnnouncementType
    };

    enum TimingReferenceType {
         NetworkTimingReference,
         AdaptiveVoice,
         FreeRun,
         OtherTimingReference
    };

    typedef boolean EchoCancellationIndType;

    struct VoiceServiceProfileAAL1Type {
         AnnouncementType announcement;
         TimingReferenceType timingReference;
         EchoCancellationIndType echoCancellationInd;
    }; // ProfileStruct for profile type = 19


    typedef long long JitterTargetType;
    typedef long long JitterBufferMaxType;

    struct VoiceServiceProfileAAL2Type {
         AnnouncementType announcement;
         TimingReferenceType timingReference;
         JitterTargetType jitterTarget;
         JitterBufferMaxType jitterBufferMax;
         EchoCancellationIndType echoCancellationInd;
    }; // ProfileStruct for profile type = 20

    struct ThresholdDataComponentType {
         MonitoredParameterType monitoredParameter; // Values defined in
         Q834Common::MonitoringParameter unsigned long long thresholdValue;
    };

    typedef sequence<ThresholdDataComponentType> ThresholdDataSeqType;

    struct ThresholdDataProfileType {
         MonitoringKindType monitoringType;
         ThresholdDataSeqType thresholdValues;
    }; // ProfileStruct for profile type = 21

    typedef sequence<ATMOverbookingFactorType> ATMOverbookingProfileType;
// ProfileStruct for profile type = 22
```

```
// Local exceptions
exception ProfileInUse {};
exception DuplicateProfileName {};

// End local definitions


interface ProfileConsumer : itut_x780::ManagedObject {
     /*
     Отображения фиксированного заголовка структурированного события:
     domain_type устанавливается на "telecommunications",
     type_name устанавливается на "ProfileEvent", а
     event_name – на приведенную ниже постоянную строку.
     Через данный интерфейс объявляются только
     новые профили.
     */

     const string profileCreation = "ProfileCreation";

     /*
     Идентификация остающихся элементов в фильтруемых данных
     структурированного события.
     */

     const string profileName = "ProfileName";
     const string profileType = "ProfileType";
     const string profileAttributeValues = "ProfileAttributeValues";

     /*
     Значения, идентифицирующие типы используемых профилей,
     приводятся ниже.
     */

     const unsigned short AAL1Profile = 1;
     const unsigned short AAL2Profile = 2;
     const unsigned short AAL2PVCProfile = 3;
     const unsigned short AAL5Profile = 4;
     const unsigned short AlarmSeverityAssignmentProfile = 5;
     const unsigned short ATMNetworkAccessProfile = 6;
     const unsigned short BridgedLANServiceProfile = 7;
     const unsigned short CESServiceProfile = 8;
     const unsigned short DS1Profile = 9;
     const unsigned short DS3Profile = 10;
     const unsigned short EthernetProfile = 11;
     const unsigned short IDLCCallProcessingProfile = 12;
     const unsigned short LESProfile = 13;
     const unsigned short MACBridgeServiceProfile = 14;
     const unsigned short SSCSParameterProfile1 = 15;
     const unsigned short SSCSParameterProfile2 = 16;
     const unsigned short TrafficDescriptorProfile = 17;
     const unsigned short UNIProfile = 18;
     const unsigned short VoiceServiceProfileAAL1 = 19;
     const unsigned short VoiceServiceProfileAAL2 = 20;
     const unsigned short ThresholdData = 21;
     const unsigned short ATMOverbookingFactorProfile = 22;

     /*
     Ниже приводится отображение в фильруемые данные в структурированном
     событии для события пользователя, которое включает создание
     объекта профиля.

     {
     {"ProfileName", any (NameType)},
     {"ProfileType", any (ProfileKindType)},
     {"EventTime", any (GeneralizedTimeType)},
     {"ProfileAttributeValues", any (ProfileStruct)},
```

```
     {"NotificationIdentifier", any (NotificationIdentifierType)}
     }

     */


}; // interface ProfileConsumer


interface ProfileUsageMgr : itut_x780::ManagedObject {

     // См. в п. 9.9.2.1 описание поведения данной операции

     void reName (
          in NameType oldProfileName,
          in NameType newProfileName)
          raises (UnknownProfiles,
               AccessDenied,
               DuplicateProfileName
               );

     // См. в п. 9.9.2.2 описание поведения данной операции

     boolean inUse (
          in NameType  profileName)
          raises (UnknownProfiles,
          AccessDenied);

     // См. в п. 9.9.2.3 описание поведения данной операции

     void suspendUse (
          in NameType profileName)
          raises (UnknownProfiles,
               AccessDenied);

     // См. в п. 9.9.2.4 описание поведения данной операции

     void resumeUse (
          in NameType profileName)
          raises (UnknownProfiles, AccessDenied) ;

     // См. в п. 9.9.2.5 описание поведения данной операции

     void deleteProfile (
          in NameType profileName)
          raises (UnknownProfiles,
               AccessDenied,
               ProfileInUse);

}; // interface ProfileUsageMgr

/*
Экземпляр данного объекта создается в программе-агенте, называемой хранили-
щем объектов профилей. Система управления поставщика является клиентом.
*/


interface ProfileRetriever : itut_x780::ManagedObject {
```

```
          // См. в п. 9.9.3.1 описание поведения данной операции

          ProfileInfoType retrieve (
               in NameType profileName)
               raises (UnknownProfiles);

     }; // interface ProfileRetriever

}; // module ProfileManager

}; // module q834_4
#endif
```

C.10 Q834Registrar.idl

```
#ifndef __Q834_4_REGISTRAR_DEFINED
#define __Q834_4_REGISTRAR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Registrar
{

     // begin definitions from other idl files


     // From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::SerialNumType SerialNumType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::AdministrationDomainType AdministrationDomainType;
     typedef Q834Common::UserLabelSeqType UserLabelSeqType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define InvalidSerialNumSyntax Q834Common::InvalidSerialNumSyntax
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define DCNTimeout Q834Common::DCNTimeout
#define DeniedAccess Q834Common::DeniedAccess
#define BackupInProgress Q834Common::BackupInProgress
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define InvalidUserLabelSyntax Q834Common::InvalidUserLabelSyntax
#define SynchInProgress Q834Common::SynchInProgress


     // End definitions from other idl files
```

```
// Local data types

// Local exceptions
exception TooManyNEs {};
exception InvalidDCNAddress {};
exception AddressLabelMismatch {};
exception APONLayerFailure {};
exception InvalidPort {};
exception HWServicesMismatch {};

// End local definitions

valuetype NERegistrarValueType: itut_x780::ManagedObjectValueType {

     public ManagedEntityIdSeqType NEList; // GET
};


interface NERegistrar : itut_x780::ManagedObject {


     // См. в п. 9.10.1.1 описание поведения данной операции

     ManagedEntityIdType registerNE (
          in DCNAddressType nEDCNAddress,
          in UserLabelType nEUserLabel,
          in AdministrationDomainType administrationDomain)
          raises (AccessDenied,
               DCNTimeout,
               AddressLabelMismatch,
               DuplicateUserLabel,
               TooManyNEs,
               InvalidDCNAddress,
               CanNotAssignManagedEntityId,
               CanNotRetrieveUserLabel,
               DeniedAccess,
               InvalidUserLabelSyntax);

     // См. в п. 9.10.1.2 описание поведения данной операции

     void modifyNEDCNAddress (
          in ManagedEntityIdType nEManagedEntityId,
          in DCNAddressType newNEDCNAddress)
          raises(AccessDenied,
               DeniedAccess,
               AddressLabelMismatch,
               DCNTimeout,
               CommFailure,
               UnknownNE,
               InvalidDCNAddress,
               BackupInProgress);



     // См. в п. 9.10.1.3 описание поведения данной операции

     ManagedEntityIdType rangeONTorONU (
          in ManagedEntityIdType oltManagedEntityId,
          in UserLabelType nEUserLabel,
          in SerialNumType serialNum,
          in ManagedEntityIdType port ) // OLT PON Port
          raises (AccessDenied,
               CommFailure,
```

```
          EquipmentFailure,
          UnknownNE,
          UnknownPort,
          MaxSubtendingNodesExceeded,
          InsufficientPONBW,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          DuplicateUserLabel,
          InvalidUserLabelSyntax,
          BackupInProgress,
          SynchInProgress );

// См. в п. 9.10.1.4 описание поведения данной операции

ManagedEntityIdType rangeReplacementNE (
     in ManagedEntityIdType oldNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType replacementSerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
          BackupInProgress,
          SynchInProgress);

// См. в п. 9.10.1.5 описание поведения данной операции

ManagedEntityIdType rangeUpgradeNE (
     in ManagedEntityIdType oldNEManagedEntityId,
     in ManagedEntityIdType newNEManagedEntityId,
     in UserLabelType newNEUserLabel,
     in SerialNumType newNESerialNum )
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidSerialNumSyntax,
          APONLayerFailure,
          EquipmentFailure,
          InvalidUserLabelSyntax,
          DuplicateUserLabel,
          HWServicesMismatch,
          BackupInProgress,
          SynchInProgress);

// См. в п. 9.10.1.6 описание поведения данной операции

ManagedEntityIdType moveONTorONU(
     in ManagedEntityIdType oldNEManagedEntityId,
     in ManagedEntityIdType newPONPort)
     raises (AccessDenied,
          CommFailure,
          UnknownPort,
          APONLayerFailure,
          EquipmentFailure,
          InsufficientPONBW,
          BackupInProgress,
          SynchInProgress,
          UnknownNE );
```

```
          // См. в п. 9.10.1.7 описание поведения данной операции

          ManagedEntityIdSeqType getSubtendingNEList(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // См. в п. 9.10.1.8 описание поведения данной операции

          ManagedEntityIdSeqType nEListGet ()
               raises (AccessDenied);


          // См. в п. 9.10.1.9 описание поведения данной операции

          void deRegisterNE(
               in ManagedEntityIdType nE)
               raises (UnknownNE,
                    AccessDenied);

          // См. в п. 9.10.1.10 описание поведения данной
          операции

          ManagedEntityIdType associateNE(
in ManagedEntityIdType preProvisionedNE,
               in ManagedEntityIdType discoveredNE)
               raises (UnknownManagedEntity,
                    AccessDenied);

     }; // interface NERegistrar

}; // module Registrar

}; // module q834_4

#endif
```

C.11 Q834ResourceAllocation.idl

```
#ifndef __Q834_4_RESOURCEALLOCATOR_DEFINED
#define __Q834_4_RESOURCEALLOCATOR_DEFINED


#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module ResourceAllocation {

     // Begin definitions from other idl files

     // From Q834Common
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::ReservationIdSeqType ReservationIdSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::EndPointType EndPointType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
```

```
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define InsufficientBW Q834Common::InsufficientBW
#define MaxSubtendingNodesExceeded Q834Common::MaxSubtendingNodesExceeded
#define UnknownNE Q834Common::UnknownNE
#define UnknownPort Q834Common::UnknownPort
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId Q834Common::UnknownReservationId
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended Q834Common::ProfileSuspended

     // End of definitions from other idl files

     // Local data types


     struct BandwidthInfoType {
          long upstreamBW;
          long downstreamBW;
     };

     struct PortBandwidthType {
          BandwidthInfoType portBandwidth;
          ManagedEntityIdType portId;
     };

     typedef sequence<PortBandwidthType> PortBandwidthSeqType;

typedef PortBandwidthSeqType AvailableSysBandwidthSeqType;
     typedef PortBandwidthSeqType ReservedBandwidthSeqType;

     struct ReservationInfoType {
          ReservationIdType reservationId;
          EndPointType endPointA;
          EndPointType endPointB;
          NameSeqType profileNameList; // servicetemplates
          ServiceInstanceIdType serviceInstanceId;
          ReservedBandwidthSeqType reservedBandwidth;
     };

     struct ReservationBandwidthType {
          ReservationIdType reservationId;
          ReservedBandwidthSeqType reservedBandwidth;
     };

     // Local exceptions

     // End local definitions


     interface ResourceAllocator : itut_x780::ManagedObject {


          // См. в п. 9.11.1.1 описание поведения данной операции

          ReservationBandwidthType reserveForService(
               in EndPointType endPointA,
               in EndPointType endPointZ,
               in NameSeqType networkCharacteristicsProfiles,
               in ServiceInstanceIdType serviceInstanceId )
               raises (UnknownNE,
                    UnknownPort,
                    UnknownProfiles,
                    InsufficientBW,
                    MaxSubtendingNodesExceeded,
```

```
                    ConnectionCountExceeded,
                    CommFailure,
                    AccessDenied,
                    ProfileSuspended );

          // См. в п. 9.11.1.2 описание поведения данной операции

          AvailableSysBandwidthSeqType cancelReservation (
               in ReservationIdType reservationId )
               raises (UnknownReservationId,
                    CommFailure,
                    AccessDenied);

          // См. в п. 9.11.1.3 описание поведения данной операции

          ReservationIdType getReservationId (
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance, AccessDenied);

          // См. в п. 9.11.1.4 описание поведения данной операции

          ReservedBandwidthSeqType reportReservedResources (
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // См. в п. 9.11.1.5 описание поведения данной операции

          ReservationIdSeqType getReservations(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE, AccessDenied);

          // См. в п. 9.11.1.6 описание поведения данной операции

          AvailableSysBandwidthSeqType cancelAllRemainingReservations(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE,
                    CommFailure,
                    AccessDenied);

          // См. в п. 9.11.1.7 описание поведения данной операции

          ReservationInfoType getReservation (
               in ReservationIdType reservationId)
               raises (UnknownReservationId,
                    AccessDenied);

          // См. в п. 9.11.1.8 описание поведения данной операции

          AvailableSysBandwidthSeqType getAvaliableSysBandwidth(
               in ManagedEntityIdType nEManagedEntityId)
               raises (UnknownNE,
                    CommFailure,
                    AccessDenied);

     }; // interface ResourceAllocator


}; // module ResourceAllocation

}; // module q834_4

#endif
```

C.12 Q834SchedulerManagement.idl

```
#ifndef __Q834_4_SCHEDULERMGR_DEFINED
#define __Q834_4_SCHEDULERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"

module q834_4 {

module SchedulerManagement {
     // Begin definitions from other idl files


     // From Q834Common

     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::OperationalStateType OperationalStateType;

#define AccessDenied Q834Common::AccessDenied
#define DuplicateUserLabel Q834Common::DuplicateUserLabel
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime

     // End definitions from other idl files

     // Local data types


     enum HourlyDailyWeeklyMonthlyIndType {
          Hourly,
          Daily,
          Weekly,
          Monthly
     };

     enum DayofWeekType {
          Sunday,
          Monday,
          Tuesday,
          Wednesday,
          Thursday,
          Friday,
          Saturday,
          Unspecified
     };

     typedef short DayOfMonthType; // 0 is interpreted to mean unspecified.


     struct TriggerTimeType
     {
          long time; // trigger time in number of seconds
          DayofWeekType dayOfWeek;
          DayOfMonthType dayofMonth;
     };

     typedef sequence<TriggerTimeType> TriggerTimeMatrixSeqType;

     struct SchedulerType
```

```
    {
         UserLabelType schedulerName;
         GeneralizedTimeType startTime; // schedule start time
         GeneralizedTimeType stopTime; // schedule stop time
         HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd;
         TriggerTimeMatrixSeqType matrix;
         OperationalStateType operationalState;
         AdministrativeStateType administrativeState;
    };

    typedef sequence<SchedulerType> SchedulerSeqType;

    // Local exceptions

    exception MatrixSchedulerTypeMismatch {};
    exception InvalidTrigger {};
    exception ScheduleInUse {};

    // End local definitions


    valuetype SchedulerMgrValueType: itut_x780::ManagedObjectValueType {

         public SchedulerSeqType schedulerList; // GET

    };

    interface SchedulerMgr : itut_x780::ManagedObject {


         // См. в п. 9.12.1.1 описание поведения данной операции

         void makeScheduler (
              in UserLabelType schedulerName,
              in GeneralizedTimeType startTime,
              in GeneralizedTimeType stopTime,
              in HourlyDailyWeeklyMonthlyIndType hourlyDailyWeeklyMonthlyInd,
              in TriggerTimeMatrixSeqType matrix)
              raises (InvalidStartTime,
                   InvalidStopTime,
                   DuplicateUserLabel,
                   MatrixSchedulerTypeMismatch,
                   AccessDenied,
                   InvalidTrigger );


         // См. в п. 9.12.1.2 описание поведения данной операции

         void suspendScheduler (
              in UserLabelType schedulerName)
              raises (UnknownScheduler,
                   AccessDenied );

         // См. в п. 9.12.1.3 описание поведения данной операции

         void resumeScheduler (in UserLabelType schedulerName) raises
(UnknownScheduler, AccessDenied );


         // См. в п. 9.12.1.4 описание поведения данной операции

         void modifyTime (
              in UserLabelType schedulerName,
              in GeneralizedTimeType newStartTime,
```

```
            in GeneralizedTimeType newStopTime)
            raises (InvalidStartTime,
                 InvalidStopTime,
                 UnknownScheduler,
                 AccessDenied );


        // См. в п. 9.12.1.5 описание поведения данной операции

        void changeSchedulerName (
            in UserLabelType oldSchedulerName,
            in UserLabelType newSchedulerName)
            raises (UnknownScheduler,
                 DuplicateUserLabel,
                 AccessDenied );

        // См. в п. 9.12.1.6 описание поведения данной операции

        void modifyTriggerTimes (
            in UserLabelType schedulerName,
            in HourlyDailyWeeklyMonthlyIndType newHourlyDailyWeeklyMonthly,
            in TriggerTimeMatrixSeqType newMatrix)
            raises (UnknownScheduler,
                 MatrixSchedulerTypeMismatch,
                 AccessDenied,
                 InvalidTrigger );

        // См. в п. 9.12.1.7 описание поведения данной операции

        void removeScheduler (
            in UserLabelType schedulerName)
            raises (UnknownScheduler,
                 AccessDenied,
                 ScheduleInUse );

        // См. в п. 9.12.1.8 описание поведения данной операции

        SchedulerType retrieveScheduler (in UserLabelType schedulerName)
            raises (UnknownScheduler,
                 AccessDenied) ;

        // См. в п. 9.12.1.9 описание поведения данной операции

        SchedulerSeqType schedulerListGet () raises (AccessDenied);

    }; // interface SchedulerMgr

}; // module SchedulerManagement

}; // module q834_4

#endif
```

C.13 Q834ServiceProvisioning.idl

```
#ifndef __Q834_4_SERVICEPROVISIONING_DEFINED
#define __Q834_4_SERVICEPROVISIONING_DEFINED


#include "Q834Common.idl"
#include "Q834ProfileManager.idl"

#pragma prefix "itu.Int"
```

```
module q834_4 {

module ServiceProvisioning {

// Begin definitions from other idl files

// From Q834Common
     typedef Q834Common::RDNType RDNType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::NameSeqType NameSeqType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::ReservationIdType ReservationIdType;
     typedef Q834Common::EndPointType EndPointType;
     typedef Q834Common::NameType NameType;
     typedef Q834Common::AdministrativeStateType AdministrativeStateType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define ConnectionCountExceeded Q834Common::ConnectionCountExceeded
#define EquipmentFailure Q834Common::EquipmentFailure
#define InsufficientBW Q834Common::InsufficientBW
#define InsufficientPONBW  Q834Common::InsufficientPONBW
#define UnknownProfiles Q834Common::UnknownProfiles
#define UnknownReservationId  Q834Common::UnknownReservationId
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance
#define ProfileSuspended ProfileManager::ProfileSuspended
#define InvalidStartTime Q834Common::InvalidStartTime
#define InvalidStopTime Q834Common::InvalidStopTime
#define ParameterViolation Q834Common::ParameterViolation
#define UnknownPort Q834Common::UnknownPort
#define ProfileSuspended Q834Common::ProfileSuspended


// End definitions from other idl files

// Local data types

// Local exceptions
     exception UnknownConnection {};
     exception ConnectionAlreadyExists {};

// End local definitions


     interface ServiceProvisioner : itut_x780::ManagedObject {


          // См. в п. 9.13.1.1 описание поведения данной операции

          ManagedEntityIdType provisionConnection(
               in EndPointType endPointA,
               in EndPointType endPointB,
               in NameSeqType networkCharacteristicsProfiles,
               in ServiceInstanceIdType serviceInstanceId,
               in AdministrativeStateType administrativeState)
               raises (UnknownNE,
                    UnknownProfiles,
                    UnknownPort,
                    InsufficientBW,
                    ConnectionCountExceeded,
                    CommFailure,
                    EquipmentFailure,
```

```
                    ParameterViolation,
                    AccessDenied,
                    InsufficientPONBW,
                    ConnectionAlreadyExists,
                    ProfileSuspended);

          // См. в п. 9.13.1.2 описание поведения данной операции

          ManagedEntityIdType provisionReservation(
               in ReservationIdType reservationId,
               in AdministrativeStateType administrativeState)
               raises (UnknownReservationId,
                    AccessDenied);

          // См. в п. 9.13.1.3 описание поведения данной операции

          void deleteConnection (
               in ManagedEntityIdType subnetworkConnectionId)
               raises (UnknownConnection,
                    CommFailure,
                    EquipmentFailure,
                    AccessDenied);

          // См. в п. 9.13.1.4 описание поведения данной операции

          ManagedEntityIdType modifyConnection (
               in ManagedEntityIdType subnetworkConnectionId,
               in ManagedEntityIdType portB,
               in NameSeqType newNetworkCharacteristicsProfiles)
               raises (UnknownConnection,
                    UnknownPort,
                    UnknownProfiles,
                    InsufficientBW,
                    AccessDenied,
                    ProfileSuspended);

          // См. в п. 9.13.1.5 описание поведения данной операции

          void suspendService (
               in ServiceInstanceIdType serviceInstanceId,
               in GeneralizedTimeType startTime,
               in GeneralizedTimeType stopTime)
               raises (UnknownServiceInstance,
                    InvalidStartTime,
                    InvalidStopTime,
                    AccessDenied);

          // См. в п. 9.13.1.6 описание поведения данной операции

          void resumeService(
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);

     }; // interface ServiceProvisioner

}; // module ServiceProvisioning

}; // module q834_4

#endif
```

C.14 Q834Synchroniser.idl

```
#ifndef __Q834_4_SYNCHRONISER_DEFINED
#define __Q834_4_SYNCHRONISER_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Synchroniser
{

// begin definitions from other idl files


// From Q834Common
     typedef Q834Common::NameType NameType;
     typedef Q834Common::UserLabelTypeUserLabelType;
     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define DCNTimeout Q834Common::DCNTimeout
#define EquipmentFailure Q834Common::EquipmentFailure
#define UnknownNE Q834Common::UnknownNE
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define BackupInProgress Q834Common::BackupInProgress
#define Timeout Q834Common::Timeout
#define SynchInProgress Q834Common::SynchInProgress


// End definitions from other idl files

// Local data types

     typedef sequence<short> CurrentListingSeqType;

     struct ScheduledSynchNEType {
          ManagedEntityIdType managedEntityId;
          UserLabelType schedulerName;
     };
     typedef sequence<ScheduledSynchNEType> ScheduledSynchNESeqType;

// Local exceptions
     exception NoSynchInProgress {};

// End local definitions

     valuetype SynchroniserValueType: itut_x780::ManagedObjectValueType {

          public ScheduledSynchNESeqType scheduledSynchNEList; // GET

     };


     interface NESynchroniser : itut_x780::ManagedObject {

          // Define constants for current event listings
```

```
const short CURRENT_ALARM = 1;
const short CURRENT_OPERATIONAL_STATE_DISABLED = 2;
const short CURRENT_ADMINSTRATIVE_STATE_LOCKED = 3;
const short CURRENT_PROTECTION_SWITCHING_EVENT = 4;
const short CURRENT_SERVICE_OUTAGE = 5;

// См. в п. 9.14.1.1 описание поведения данной операции

void synchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          EquipmentFailure,
          BackupInProgress,
          SynchInProgress);

// См. в п. 9.14.1.2 описание поведения данной операции

void abortSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          EquipmentFailure,
          NoSynchInProgress);

// См. в п. 9.14.1.3 описание поведения данной операции

void scheduleSynchNE(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// См. в п. 9.14.1.4 описание поведения данной операции

void modifyNESynchSchedule(
     in ManagedEntityIdType nEManagedEntityId,
     in UserLabelType newSchedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler);

// См. в п. 9.14.1.5 описание поведения данной операции

void cancelScheduledSynchNE(
     in ManagedEntityIdType nEManagedEntityId)
     raises (AccessDenied,
          UnknownNE);

// См. в п. 9.14.1.6 описание поведения данной операции

void synchCurrentEventListings(
     in ManagedEntityIdType nEManagedEntityId,
     in CurrentListingSeqType currentListingTypeList)
     raises (AccessDenied,
          CommFailure,
          DCNTimeout,
          UnknownNE,
```

```
                    EquipmentFailure,
                    Timeout);

          // См. в п. 9.14.1.7 описание поведения данной операции

          ScheduledSynchNESeqType scheduledSynchNEListGet ()
               raises (AccessDenied);

     }; // interface NESynchroniser

}; // module Synchroniser

}; // module q834_4

#endif
```

C.15 Q834Test.idl

```
#ifndef __Q834_4_TEST_DEFINED
#define __Q834_4_TEST_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module Test
{

     // begin definitions from other idl files

     // From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::ManagedEntityIdSeqType ManagedEntityIdSeqType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::LoopbackLocationIdSeqType
     LoopbackLocationIdSeqType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::ServiceInstanceIdType ServiceInstanceIdType;
     typedef Q834Common::TrackingObjectIdType TrackingObjectIdType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define InvalidScheduler Q834Common::InvalidScheduler
#define InvalidStopTime Q834Common::InvalidStopTime
#define InvalidStartTime Q834Common::InvalidStartTime
#define UnknownManagedEntity Q834Common::UnknownManagedEntity
#define UnknownNE Q834Common::UnknownNE
#define UnknownServiceInstance Q834Common::UnknownServiceInstance

     // End definitions from other idl files


     // Local data types
```

```
    enum DirectionalityType {
           Egress,
        Ingress,
        BothDirections
    };

    struct ATMLoopbackInfoType {
        DirectionalityType directionality;
        LoopbackLocationIdSeqType targetLLID;
        boolean segmentCellInd;
    };

    typedef  unsigned short TestIterationNumType;

    struct ATMLoopbackResultType {
        LoopbackLocationIdSeqType loopbackingLLID;
        unsigned long responseTime; //in microseconds
        boolean succeeded; //true or false
    };

    typedef sequence<ATMLoopbackResultType> ATMLoopbackResultSeqType;

    struct AggregateATMLoopbackResultType {
        unsigned short iterationSeqNum;
        ATMLoopbackResultSeqType iterationTestResults;
    };

    typedef sequence<AggregateATMLoopbackResultType>
    AggregateATMLoopbackResultSeqType;

    struct ATMContinuityCheckInfoType {
        DirectionalityType directionality;
        boolean segmentCellInd;
    };


    typedef short DiagnosticType; // Supplier specific.
    typedef DiagnosticType ResourceSelfTestInfoType;
    typedef sequence<ResourceSelfTestInfoType> ResourceSelfTestInfoSeqType;

    struct ResourceSelfTestResultType {
           DiagnosticType diagnostic;
           boolean testPassed;
    };

    typedef sequence<ResourceSelfTestResultType>
    ResourceSelfTestResultSeqType;
    typedef long long CCSetUpIdType;
    typedef TrackingObjectIdType TestTrackingObjectIdType;
    typedef long long LoopbackTrackingObjectIdType;


    typedef unsigned short LoopbackTestType;

    struct LoopbackInfoType {
        LoopbackTrackingObjectIdType trackingId;
        unsigned long remainingTime; // in seconds
ManagedEntityIdType ctpId;
        LoopbackTestType loopbackTest;
        DirectionalityType directionality;
    };

    typedef sequence<LoopbackInfoType> LoopbackInfoSeqType;
```

```
typedef string SupportedTestType;

typedef sequence<SupportedTestType> SupportedTestSeqType;

struct ScheduledTestNEType {
       ManagedEntityIdType managedEntityId;
       SupportedTestType supportedTest;
       UserLabelType schedulerName;
};
typedef sequence<ScheduledTestNEType> ScheduledTestNESeqType;

struct TestHistoryType {
     ManagedEntityIdType managedEntityId;
     SupportedTestType supportedTest;
     StatusValueType testStatus;
};
typedef sequence<TestHistoryType> TestHistorySeqType;

// Local exceptions

exception InvalidTimeoutPeriod {};
exception NotAvailableForTest {};
exception InvalidLocationId {};
exception UnknownTest {};
exception UncontrolledTestInProgress {};
exception InvalidDirection {};
exception InvalidTestOperations {};

// End local definitions

valuetype TestActionPerformerValueType: itut_x780::ManagedObjectValueType {

     public ScheduledTestNESeqType scheduledTestNEList; // GET

};


interface TestActionPerformer : itut_x780::ManagedObject {

     // См. в п. 9.15.1.1 описание поведения данной операции

     AggregateATMLoopbackResultSeqType aTMLoopback (
          in UserIdType testRequestorId,
          in ManagedEntityIdType ctp,
          in ATMLoopbackInfoType aTMLoopbackInfo,
          in TestIterationNumType testIterationNum,
          in ServiceInstanceIdType serviceInstanceId)
          raises (AccessDenied,
               CommFailure,
               UnknownManagedEntity,
               NotAvailableForTest,
               InvalidLocationId,
               InvalidDirection);

     // См. в п. 9.15.1.2 описание поведения данной операции

     CCSetUpIdType initializeContinuityCheck(
          in UserIdType testRequestorId,
          in ManagedEntityIdType sourceCtp,
          in ATMContinuityCheckInfoType aTMContinuityCheckInfo,
          in GeneralizedTimeType stopTime,
          in ServiceInstanceIdType serviceInstanceId)
          raises (AccessDenied,
               CommFailure,
```

```
          UnknownManagedEntity,
          NotAvailableForTest,
          InvalidStartTime,
          InvalidStopTime,
          InvalidDirection);

// См. в п. 9.15.1.3 описание поведения данной операции

void terminateContinuityCheck(
     in CCSetUpIdType cCSetUpId)
     raises (AccessDenied,
          CommFailure,
          UnknownTest);

// См. в п. 9.15.1.4 описание поведения данной операции

TestTrackingObjectIdType scheduleResourceSelfTest(
     in UserIdType testRequestorId,
     in ManagedEntityIdType targetNE,
     in unsigned long timeOutPeriod, //In seconds.
     in ResourceSelfTestInfoSeqType specificTestInfo,
     in UserLabelType schedulerName)
     raises (AccessDenied,
          UnknownNE,
          UnknownScheduler,
          InvalidScheduler,
          InvalidTimeoutPeriod,
          InvalidTestOperations);

// См. в п. 9.15.1.5 описание поведения данной операции

         void modifyResourceSelfTestSchedule(
           in TestTrackingObjectIdType testTrackingObjectId,
           in UserLabelType newSchedulerName)
           raises (AccessDenied,
                UnknownTest,
                UnknownScheduler,
                InvalidScheduler);

// См. в п. 9.15.1.6 описание поведения данной операции

     void cancelScheduledResourceSelfTest (
          in TestTrackingObjectIdType testTrackingObjectId)
          raises (AccessDenied,
               UnknownTest);

// См. в п. 9.15.1.7 описание поведения данной операции

TestTrackingObjectIdType conductResourceSelfTest (
     in UserIdType testRequestorId,
     in ManagedEntityIdType targetNE,
     in unsigned long timeOutPeriod, //In seconds.
     in ResourceSelfTestInfoSeqType specificTestInfo)
     raises (AccessDenied,
          CommFailure,
          UnknownNE,
          InvalidTimeoutPeriod,
          InvalidTestOperations);

// См. в п. 9.15.1.8 описание поведения данной операции

ResourceSelfTestResultSeqType terminateResourceSelfTest (
     in TestTrackingObjectIdType testTrackingObjectId)
     raises (UnknownTest,
```

```
          UncontrolledTestInProgress,
          AccessDenied);

// См. в п. 9.15.1.9 описание поведения данной операции

LoopbackTrackingObjectIdType initiateLoopback (
     in UserIdType testRequestorId,
     in ManagedEntityIdType loopingCtp,
     in long duration, //In minutes.
     in DirectionalityType directionality,
     in LoopbackTestType loopbackTest,
     in ServiceInstanceIdType serviceInstanceId)
     raises (AccessDenied,
          CommFailure,
          UnknownManagedEntity,
          NotAvailableForTest);

// См. в п. 9.15.1.10 описание поведения данной операции

void terminateLoopback(
     in LoopbackTrackingObjectIdType testTrackingObjectId)
     raises (UnknownTest,
          AccessDenied);

// См. в п. 9.15.1.11 описание поведения данной операции

LoopbackInfoType getLoopbackInfo(
     in ManagedEntityIdType cTP)
     raises (UnknownManagedEntity,
          AccessDenied);

// См. в п. 9.15.1.12 описание поведения данной операции

LoopbackInfoSeqType getLoopbackInfoByNE(
     in ManagedEntityIdType nEId)
     raises (UnknownManagedEntity,
          AccessDenied);

// См. в п. 9.15.1.13 описание поведения данной операции

StatusValueType getTestStatus (
     in LoopbackTrackingObjectIdType id)
     raises (AccessDenied,
          UnknownTest);

// См. в п. 9.15.1.14 описание поведения данной операции

ScheduledTestNESeqType  scheduledTestNEListGet()
     raises (AccessDenied);

// См. в п. 9.15.1.15 описание поведения данной операции

TestHistorySeqType testHistoryByManagedEntity (
     in ManagedEntityIdType managedEntityId)
     raises (UnknownManagedEntity,
          AccessDenied);
```

```
          // См. в п. 9.15.1.16 описание поведения данной операции

          TestHistorySeqType testHistoryByServiceInstance (
               in ServiceInstanceIdType serviceInstanceId)
               raises (UnknownServiceInstance,
                    AccessDenied);

     }; // interface TestActionPerformer

}; // module Test

}; // module q834_4

#endif
```

C.16 Q834Filetransfer.idl

```
#ifndef __Q834_4_TRANSFERMGR_DEFINED
#define __Q834_4_TRANSFERMGR_DEFINED

#include "Q834Common.idl"

#pragma prefix "itu.Int"


module q834_4
{

module FileTransfer
{

// begin definitions from other idl files

// From Q834Common

     typedef Q834Common::ManagedEntityIdType ManagedEntityIdType;
     typedef Q834Common::UserLabelType UserLabelType;
     typedef Q834Common::StatusValueType StatusValueType;
     typedef Q834Common::DCNAddressType DCNAddressType;
     typedef Q834Common::FilenameType FilenameType;
     typedef Q834Common::TransferTrackingObjectIdType
     TransferTrackingObjectIdType;
     typedef Q834Common::UserIdType UserIdType;
     typedef Q834Common::PasswordType PasswordType;
     typedef Q834Common::GeneralizedTimeType GeneralizedTimeType;

#define AccessDenied Q834Common::AccessDenied
#define CommFailure Q834Common::CommFailure
#define UnknownScheduler Q834Common::UnknownScheduler
#define UnknownDestinationServer Q834Common::UnknownDestinationServer
#define InvalidScheduler Q834Common::InvalidScheduler
#define UnknownRecordSet Q834Common::UnknownRecordSet

// End definitions from other idl files

// Local data types



     struct FileTransferHistoryType {
       ManagedEntityIdType recordSetId;
       DCNAddressType destinationServerAddr;
       UserIdType  userId;
```

```
      FilenameType destinationFile;
      GeneralizedTimeType transferTime;
    };
    typedef sequence<FileTransferHistoryType> FileTransferHistorySeqType;

    struct ScheduledFileTransferType {
      ManagedEntityIdType recordSetId;
      UserLabelType schedulerName;
    };
    typedef sequence<ScheduledFileTransferType> ScheduledFileTransferSeqType;

// Local exceptions

    exception UnknownTransferProcess {};

// End local definitions


    valuetype TransferMgrValueType: itut_x780::ManagedObjectValueType {

        public FileTransferHistorySeqType fileTransferHistoryList; // GET
        public ScheduledFileTransferSeqType scheduledFileTransferList; // GET

    };

    interface TransferMgr : itut_x780::ManagedObject {

        // См. в п. 9.16.1.1 описание поведения данной операции

            TransferTrackingObjectIdType fileTransfer(
                in ManagedEntityIdType recordSetId,
                in DCNAddressType destinationServerAddr,
                in UserIdType userId,
                in PasswordType password,
                in FilenameType destinationFile,
                in boolean overwriteExistingFile)
                raises (AccessDenied,
                    CommFailure,
                    UnknownRecordSet,
                    UnknownDestinationServer
                    );


        // См. в п. 9.16.1.2 описание поведения данной операции

            TransferTrackingObjectIdType scheduleFileTransfer(
                in ManagedEntityIdType recordSetId,
                in DCNAddressType destinationServerAddr,
                in UserIdType userId,
                in PasswordType password,
                in FilenameType destinationFile,
                in boolean overwriteExistingFile,
                in UserLabelType schedulerName)
                raises (AccessDenied,
                    UnknownRecordSet,
                    UnknownDestinationServer,
                    UnknownScheduler,
                    InvalidScheduler);

        // См. в п. 9.16.1.3 описание поведения данной операции

              void modifyFileTransferSchedule(
                in TransferTrackingObjectIdType transferTrackingObjectId,
                in UserLabelType newSchedulerName)
```

```
                    raises (AccessDenied,
                         UnknownTransferProcess,
                         UnknownScheduler,
                         InvalidScheduler);

               // См. в п. 9.16.1.4 описание поведения данной операции

                    void cancelScheduledFileTransfer (
                         in TransferTrackingObjectIdType transferTrackingObjectId)
                         raises (AccessDenied,
                              UnknownTransferProcess);

               // См. в п. 9.16.1.5 описание поведения данной операции

                    StatusValueType getStatus(
                         in TransferTrackingObjectIdType transferTrackingObjectId)
                         raises (UnknownTransferProcess,
                              AccessDenied);

               // См. в п. 9.16.1.6 описание поведения данной операции

               FileTransferHistorySeqType fileTransferHistoryListGet ()
                    raises (AccessDenied);

               // См. в п. 9.16.1.7 описание поведения данной операции

               ScheduledFileTransferSeqType scheduledFileTransferListGet()
                    raises (AccessDenied);

          }; // interface TransferMgr

}; // module FileTransfer

}; // module q834_4

#endif
```

Приложение D

Пример шаблонов конечной точки

В таблицах D.1 и D.2 приводятся примеры шаблонов того, как может быть определена конечная точка в зависимости от типа службы. В таблице D.1 показаны примеры для случая, когда конечная точка является частью порта NNI. В таблице D.2 показаны примеры для случая, когда конечная точка является частью порта UNI.

Таблица D.1/Q.834.4 – Конечные точки порта NNI

Тип службы	Порт	Параметр	Профили
DS1	TDM DS3	Канал #	–
DS3	TDM DS3	–	–
DS1	ATM DS3	VPI/VCI	–
DS3	ATM DS3	VPI/VCI	–
DS1	TDM OC-n	STS #/VT1.5 #	–
DS1	ATM OC-n	VPI/VCI	–
DS3	ATM OC-n	VPI/VCI	–
LAN с мостом	ATM OC-n	VPI/VCI	–
LAN с мостом	ATM DS3	VPI/VCI	–
LAN с мостом	TDM DS3	канал #	–
LAN с мостом	TDM OC-n	STS #/VT1.5 #	–
Речевая	ATM DS3	VPI/VCI CID	TrafficDescriptorProfile (входящий) TrafficDescriptorProfile (исходящий)
Речевая	–	Интерфейсная группа #/ Эталонное значение вызова	–
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (входящий) TrafficDescriptorProfile (исходящий)

Таблица D.2/Q.834.4 – Конечные точки порта UNI

Тип службы	Порт	Параметр	Профили
DS1	TDM DS3	Канал #	DS1Profile CESProfile AAL1Profile
DS3	TDM DS3	–	DS3Profile CESProfile AAL1Profile
DS1	ATM DS3	VPI/VCI	–
DS3	ATM DS3	VPI/VCI	–
DS1	TDM OC-n or STS-n	STS #/VT1.5 #	AAL1Profile CESProfile
DS1	ATM OC-n	VPI/VCI	–
DS3	ATM OC-n	VPI/VCI	–
LAN с мостом	Ethernet	Логический порт #	BridgedLANServiceProfile AAL5Profile или AAL1Profile
Речевая	RJ-11	–	VoiceServiceProfileAAL2 SSCSParameterProfile1 SSCSParameterProfile2 LESService
Речевая	RJ-11	–	VoiceServiceProfileAAL1
ATM	ATM	VPI/VCI	TrafficDescriptorProfile (входящий) TrafficDescriptorProfile (исходящий)
VLAN	Ethernet	Тег VLAN	–

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

Серия A	Организация работы МСЭ-Т
Серия B	Средства выражения: определения терминов, символы, классификация
Серия C	Общая статистика электросвязи
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	TMN и техническое обслуживание сетей: международные системы передачи, телефонные, телеграфные, факсимильные и арендованные каналы
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 и взаимосвязь открытых систем
Серия Y	Глобальная информационная инфраструктура и аспекты межсетевого протокола (IP)
Серия Z	Языки и общие аспекты программного обеспечения для систем электросвязи

26846

Отпечатано в Швейцарии
Женева, 2005 г.