ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО 9506-2— 2014

Системы промышленной автоматизации и интеграция

СПЕЦИФИКАЦИЯ ПРОИЗВОДСТВЕННЫХ СООБЩЕНИЙ

Часть 2

Спецификация протокола

ISO 9506-2:2003
Industrial automation systems and integration — Manufacturing Message Specification — Part 2: Protocol specification
(IDT)

Издание официальное

Москва
Стандартинформ
2015

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН ООО «НИИ экономики связи и информатики «Интерэкомс» (ООО «НИИ «Интерэкомс») на основе собственного аутентичного перевода на русский язык международного стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 100 «Стратегический и инновационный менеджмент»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 26 ноября 2014 г. № 1864-ст

4 Настоящий стандарт идентичен международному стандарту ИСО 9506-2:2003 «Системы промышленной автоматизации и интеграция. Спецификация производственных сообщений. Часть 2. Спецификация протокола» (ISO 9506-2:2003 «Industrial automation systems — Manufacturing Message Specification — Part 2: Protocol specification»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в ГОСТ Р 1.0–2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте национального органа Российской Федерации по стандартизации в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2015

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения 1
2 Нормативные ссылки 1
3 Термины и определения 2
4 Сокращения 5
5 Соглашения 5
6 Элементы протокольной процедуры 9
7 Блоки данных протокола спецификации производственных сообщений MMS PDU 15
8 Среда и протокол общего управления 59
9 Протокол ответа на услугу, удовлетворяющую заданным требованиям 63
10 Протокол поддержки VMD 66
11 Протокол управления доменом 70
12 Протокол управления активизацией программы 76
13 Протокол управления блоком 82
14 Протокол доступа к переменной 88
15 Протокол обмена данными 100
16 Протокол управления семафором 101
17 Протокол связи с оператором 104
18 Протокол управления событием 105
19 Протокол условий события 110
20 Протокол действия события 114
21 Протокол регистрации события 116
22 Протокол перечня условий события 121
23 Протокол управления журналом 125
24 Отображение на нижележащие услуги связи 128
25 Утверждение и конфигурация инициализации 131
Приложение А (обязательное) Связь *M*-услуг с сервисным элементом управления ассоциацией (ACSE) и услугами представления данных 148
Приложение В (обязательное) Абстрактный формат конфигурации и инициализации 151
Приложение С (обязательное) Протокол доступа к файлу 164
Приложение D (справочное) Протокол управления файлом 166
Приложение Е (справочное) Рассеянный доступ 169
Приложение F (справочное) Тип данных REAL 170
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации 171

Введение

Настоящий стандарт определяет услуги для различных разработанных устройств и блоков системы управления производственным процессом. Настоящий стандарт может использоваться как самостоятельно, так и вместе с другими стандартами, описывающими применение подмножества указанных услуг в части определенных типов устройств.

Услуги, обеспечиваемые спецификацией производственных сообщений (MMS), могут быть разными: от простых до самых сложных. Совсем не обязательно, чтобы все указанные услуги поддерживались всеми устройствами. В некоторых случаях поддерживаемое подмножество ограничивается сопутствующими стандартами. Во всех случаях данное множество может быть ограниченно пользователем. Важнейшие характеристики поддерживаемого подмножества услуг приведены далее:

a) применимость конкретной услуги к конкретному устройству;

b) сложность услуг и накладываемые требования;

c) сложность обеспечения конкретного класса услуг сети и сложность самого устройства.

Требования безопасности

При практической реализации MMS в защищенных (безопасных) приложениях необходимо обеспечить выполнение требований архитектуры безопасности модели взаимосвязи открытых систем (модели OSI). Настоящий стандарт определяет простые средства аутентификации (пароли) и управления доступом. Системы более высокой степени безопасности нуждаются в дополнительных требованиях по безопасности, выходящих за пределы области применения настоящего стандарта. Настоящий стандарт не гарантирует отсутствие отказов в рамках установленных обязательств.

Сложность услуг и требований

Некоторые MMS-услуги представляют собой достаточно сложные функции. Устройства, используемые в простых приложениях, часто не требуют таких сложных функций и не поддерживают соответствующие MMS-услуги.

Ключевые слова

Межсетевое взаимодействие приложения
Протокол прикладного уровня
Система обработки информации
Производственная коммуникационная сеть
Спецификация производственных сообщений
Система цифрового управления
Взаимосвязь открытых систем
Ссылочная модель OSI
Система управления процессом
Программируемый контроллер
Программируемое устройство
Система управления робототехническим оборудованием
Виртуальное производственное устройство

Общие положения

Настоящий стандарт — это один из серии стандартов, обеспечивающих взаимосвязь систем обработки информации. Он позиционируется на прикладном уровне среды взаимосвязи открытых систем по отношению к другим стандартам в качестве прикладного сервисного элемента (ASE) базовой ссылочной модели обеспечения взаимосвязи открытых систем (см. ИСО 7498).

Целью обеспечения взаимосвязи открытых систем является организация (с минимальными техническими соглашениями за рамками соответствующих стандартов) взаимосвязи систем обработки информации:

a) различных изготовителей;

b) различных форм управления;

c) различных уровней сложности;

d) различных временных интервалов.

Цель

Целью настоящего стандарта является определение спецификаций производственных сообщений. Настоящий стандарт тесно связан с областью применения определения спецификации услуги производственных сообщений (см. ИСО 9506-1). Он использует услуги, обеспечиваемые коммуникационной системой, которая предполагается для передачи элементов PDU.

Протокол MMS структурируется так, чтобы можно было определить подмножества протокола. Варианты и доступные опции настоящего стандарта делают возможным использование спецификаций производственных сообщений в большинстве приложений. Таким образом, практические реализации, удовлетворяющие установленным в настоящем стандарте требованиям в минимальной степени, не годятся для всех возможных обстоятельств. Важно квалифицировать все ссылки на настоящий стандарт, учитывая особенности рассматриваемых опций и условия их практической значимости.

Примечание — Услуги, описанные в настоящем стандарте, являются универсальными. Они составляют базу для сопутствующих стандартов с конкретными классами приложений. Услуги настоящего стандарта также могут быть задействованы обособленно (без использования сопутствующих стандартов).

Необходимо отметить, что некоторые действительные протокольные последовательности являются очень большими. Поэтому не представляется возможным (используя современную технологию) верифицировать, что некоторая практическая реализация работает с протоколом, определенным в настоящем стандарте, корректно при всех обстоятельствах. Путем тестирования можно достоверно установить, что рассматриваемая практическая реализация задействует корректно данный протокол на показательно выбранных обстоятельствах.

Издание

Настоящий стандарт отличается от первого издания ИСО 9506-2 тем, что в нем исправлены ошибки в протоколах, связанных с определениями типов ASN.1 и моделированием структур. В настоящем стандарте также исправлены несколько типографских ошибок.

Отличия настоящего стандарта от ИСО/МЭК 9506-2:1990 заключаются в том, что:

a) материалы ИСО/МЭК ТО 13345, описывающие подмножества протокола MMS, включены в настоящий стандарт;

b) все материалы изменений 1 и 2 включены в настоящий стандарт как технические поправки;

c) формальная модель объекта, используемая в ИСО 9506-1, обеспечивает некоторые типы определений для протокола, описанного в настоящем стандарте. Так, утверждение IMPORT включено в модуль ASN.1;

d) услуги и протокол, ранее опубликованные в сопутствующих стандартах (ИСО/МЭК 9506-3, ИСО/МЭК 9506-4 и ИСО/МЭК 9506-6), включены в базовый стандарт.

В результате указанных изменений нет необходимости использовать отдельный абстрактный синтаксис для каждого сопутствующего стандарта. Все сопутствующие стандарты теперь могут ссылаться на единый абстрактный синтаксис базового стандарта. При этом использование других абстрактных синтаксисов остается возможным для обеспечения совместимости. Отпала необходимость и в отдельном определении модуля в разделе 19 первого издания ИСО/МЭК 9506-2 (указанный раздел удален);

e) требования к связи MMS обобщены, и MMS описан с учетом абстрактного множества услуг, необходимых для их поддержки. Соотношение между указанным абстрактным множеством услуг и услугами, обеспечиваемыми семейством протоколов связи модели OSI, приведено в приложении. Это дает возможность обеспечить корректную работу MMS с альтернативными системами связи (уменьшение необходимого объема стековой памяти) путем использования эквивалентности указанных абстрактных услуг;

f) ограничения на обозначения, которые могут быть использованы как идентификаторы, ослаблены. Теперь идентификатор может начинаться с цифры и, как расширение, может состоять только из цифр;

g) многие (но не все) реализации **VisibleString** заменены на новую реализацию **MMSString,** которая дает возможность использовать произвольную строку символов, определенных в ИСО 10646. Аналогично, эти более общие строки также могут быть использованы как идентификаторы. Добавленные новые параметры СВВ создают условия для обсуждения условий применения указанных более общих строк;

h) введена новая услуга **ReconfigureProgramInvocation** (активизация программы реконфигурации), вставленная в раздел по управлению программами активизации. Данная услуга задает технологию динамического изменения составляющих доменов рассматриваемой вызова программы;

i) добавлена новая область к объектной модели поименованной переменной и поименованного типа. Данная область может быть использована для описания семантики, ассоциированной с поименованной

переменной или с поименованным типом. Эта область определена предварительно либо ее значение установлено как имя поименованного типа (используемого для его построения в услуге определения поименованной переменной **DefineNamedVariable** или в услуге определения поименованного типа **DefineNamedType**). Настоящая область указана вместе с услугой получения атрибутов доступа к переменной **GetVariableAccessAttributes** или услугой получения атрибутов поименованного типа **GetNamedTypeAttributes,** если оговорена величина **sem** (нового параметра СВВ);

j) текст стандарта отредактирован таким образом, что разделов стало больше, и они стали короче;

k) тип данных **Real** удален из настоящего стандарта;

l) из текста настоящего стандарта удалены материалы по рассеянному доступу. Они размещены в справочном приложении;

m) в соответствии с рекомендациями ИСО/МЭК 8824-1 все утверждения **EXTERNAL** в протоколе заменены утверждениями **CHOICE {EXTERNAL, EMBEDDED PDV}**;

n) сущность PICS первого издания заменена разделом, содержащим информацию о конфигурации и инициализации. Настоящий раздел содержит указания по инициализаци в некоторых областях (их относительно немного) для VMD и подчиненных объектов, а также обеспечивает отчет (в форме таблицы) о значениях инициализации в других областях (в соответствии с возможностями разработчика). Добавлено новое приложение (см. приложение В). Оно содержит описание модуля ASN.1, используемого при передаче информации, содержащейся в указанной таблице.

Протокол

В настоящее время в результате использования методики моделирования объекта ASN.1 протокол существует в виде трех отдельных модулей. Один модуль является частью объектной модели, описанной в ИСО 9506-1. Два других модуля определены в настоящем стандарте, содержащем описания контента и структуры всех корректных блоков данных PDU. Несмотря на то что формулировка ASN.1 может быть различной в некоторых случаях, сущности PDU, полученные с помощью приложений, описанных в первом издании ИСО 9506, идентичны соответствующим сущностям настоящего издания. По этой причине настоящий стандарт по-прежнему идентифицируют в качестве основной версией № 1 (номера других версий изменены, чтобы соответствовать всем новым добавлениям к настоящему стандарту).

Необходимо указать два исключения:

1) синтаксические расширения, определенные сопутствующими стандартами, теперь идентифицированы новыми параметрами СВВ вместо отдельного абстрактного синтаксиса. Следовательно, для любых возможностей сопутствующих стандартов, использующих MMS, имеет место изменение в пусковом PDU. При этом если указанные возможности сопутствующего стандарта не используются, то пусковой PDU остается тем же, что определен в первом издании;

2) некоторые небольшие изменения внесены в теговую разметку услуги **ChangeAccessControl** (управление изменением доступа, часть изменения 2) для приведения ее в соответствие с протоколом услуги **GetNameList** (получение перечня имен) и услуги **Rename** (переименование).

Кодирование PDU с помощью PER (см. ИСО/МЭК 8825-2) может быть не полностью совместимо с PDU, генерированным в первом издании ИСО/МЭК 9506. Дело в том, что замена некоторого типа сущностью CHOICE (выбор), содержащей данный тип, приводит к различным вариантам кодирования на базе PER. То же самое имеет место при BER-кодировании для рассматриваемых двух ситуаций. Таким образом, если PDU содержит элементы типа **EXTERNAL**, то они будут заменены сущностью CHOICE (выбор), используемой при PER-кодировании.

Модуль ASN.1

Модули ASN.1, определенные в ИСО 9506, можно получить в секретариате ПК4 в электронном формате. Модули доступны в двух видах: 1) в опубликованном виде; 2) с удаленными скобками типа IF-ENDIF.

Указанные файлы доступны на сайте: gost.e-ecolog.ru

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы промышленной автоматизации и интеграция

СПЕЦИФИКАЦИЯ ПРОИЗВОДСТВЕННЫХ СООБЩЕНИЙ
Часть 2

Спецификация протокола

Industrial automation systems and integration. Manufacturing message specification. Part 2. Protocol specification

Дата введения — 2016—01—01

1 Область применения

Спецификация производственных сообщений — это стандарт прикладного уровня, поддерживающего связь посредством сообщений, отправляемых на (получаемых с) программируемые устройства среды компьютерно-интегрированного производства (CIM).

1.1 Спецификации

Настоящий стандарт содержит описания:

a) процедуры одинарного протокола передачи данных и управляющей информации из одной сущности приложения в другую одноранговую прикладная сущность в рассматриваемом MMS-контексте;

b) средств выбора услуг для сущностей приложения при обеспечении связи в MMS-контексте;

c) структуры блока данных протокола спецификации обмена производственными сообщениями, используемого для передачи данных и управляющей информации.

1.2 Процедуры

Указанные выше процедуры определены в терминах:

a) взаимодействий одноранговых сущностей приложения путем обмена блоками данных протокола спецификации обмена производственными сообщениями;

b) взаимодействий MMS-провайдеров и MMS-пользователей в той же системе путем обмена MMS-примитивами;

c) взаимодействий MMS-провайдеров и абстрактных услуг, предоставляемых нижележащими системами связи.

1.3 Применимость

Указанные процедуры применимы к экземплярам связи между системами, поддерживающими MMS внутри прикладного уровня ссылочной модели OSI, и системами, для которых необходима возможность обеспечения взаимосвязи в среде взаимодействия открытых систем.

1.4 Соответствие

Настоящий стандарт также содержит описание требований соответствия систем, реализующих указанную процедуру, но при этом отсутствуют описания проверок, демонстрирующих соответствие указанным требованиям.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при применении настоящего стандарта. В случае ссылок на документы, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В том случае, когда дата утверждения не приведена, следует пользоваться последней редакцией ссылочных документов, включая любые поправки и изменения к ним.

Издание официальное

ИСО/МЭК 646 Информационные технологии. 7-битный набор кодированных символов ИСО для обмена информацией (ISO/IEC 646 Information technology — ISO 7-bit coded character set for information interchange)

ИСО/МЭК 7498-1 Информационные технологии. Взаимодействие открытых систем. Базовая эталонная модель. Часть 1. Базовая модель (ISO/IEC 7498-1 Information technology — Open Systems Interconnection — Basic reference model. Part 1: The basic model)

ИСО 7498-2 Системы обработки информации. Взаимодействие открытых систем. Базовая эталонная модель. Часть 2. Архитектура защиты (ISO 7498-2 Information processing systems — Open Systems Interconnection; basis reference model; Part 2: Security architecture)

ИСО/МЭК 7498-3 Информационные технологии. Взаимодействие открытых систем. Базовая эталонная модель. Часть 3. Присвоение имен и адресация (ISO/IEC 7498-3 Information technology — Open Systems Interconnection — Basic reference model. Part 3: Naming and addressing)

ИСО 8571 (все части) Системы обработки информации. Взаимодействие открытых систем. Передача, доступ и управление файлами (ISO 8571 Information processing systems; Open Systems Interconnection; file transfer, access and management)

ИСО/МЭК 8650-1 Системы обработки информации. Машинная графика. Привязки к языку базовой графической системы (GKS). Часть 1. ФОРТРАН (ISO/IEC 8650-1 Information processing systems; computer graphics; graphical kernel system (GKS) language bindings; part 1: FORTRAN)

ИСО/МЭК 8822 Информационные технологии. Взаимосвязь открытых систем. Определение службы представления данных (ISO/IEC 8822 Information technology — Open Systems Interconnection — Presentation service definition)

ИСО/МЭК 8824-1 Информационные технологии. Нотация абстрактного синтаксиса версии 1 (ASN.1). Часть 1. Спецификация базовой нотации (ISO/IEC 8824-1 Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation)

ИСО/МЭК 8824-2 Информационные технологии. Нотация абстрактного синтаксиса один (ASN.1). Часть 2. Спецификация информационных объектов (ISO/IEC 8824-2 Information technology — Abstract Syntax Notation One (ASN.1): Information object specification)

ИСО/МЭК 8825-1 Информационные технологии. Правила кодирования ASN.1. Часть 1. Спецификация основных (BER), канонических (CER) и различительных правил кодирования (DER) (ISO/IEC 8825-1 Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER))

ИСО/МЭК 8825-2 Информационные технологии. Правила кодирования ASN.1. Часть 2. Спецификация правил уплотненного кодирования (PER) (ISO/IEC 8825-2, Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER))

ИСО 9506-1 Системы промышленной автоматизации. Спецификация производственных сообщений. Часть 1. Определение услуг (ISO 9506-1, Industrial automation systems — Manufacturing Message Specification — Part 1: Service definition)

ИСО/МЭК 9545 Информационные технологии. Взаимосвязь открытых систем. Структура прикладного уровня (ISO/IEC 9545, Information technology — Open Systems Interconnection — Application Layer structure)

ИСО/МЭК 10731 Информационные технологии. Взаимосвязь открытых систем. Базовая эталонная модель. Соглашения, касающиеся определения услуг в OSI (ISO/IEC 10731, Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services)

ANSI/IEEE 754 Стандарт IEEE на бинарную арифметику с плавающей точкой (ANSI/IEEE 754, Binary floating-point arithmetic)

3 Термины и определения

В настоящем стандарте использованы следующие термины с соответствующими определениями.

Примечание — Определения, содержащиеся в настоящем разделе, используют сокращения, приведенные в разделе 4.

3.1 Определения ссылочных моделей

Настоящий стандарт базируется на понятиях, разработанных в Базовой эталонной модели взаимосвязи открытых систем (см. ИСО 7498). В настоящем стандарте использованы следующие термины:

a) прикладная сущность;
b) прикладной процесс;
c) прикладной сервисный элемент;
d) открытая система;
e) (*N*) — протокол;
f) (*N*) — блок данных протокола;
h) (*N*) — уровень;
i) система.

3.2 Определение соглашения об услугах

Настоящий стандарт использует следующие термины, определенные конвенциями по определению услуг OSI (см. ИСО/МЭК 10731). Эти термины применяются для спецификации производственных сообщений:

a) подтверждение;
b) индикация;
c) примитив;
d) запрос;
e) ответ;
f) примитив услуг;
g) провайдер услуг;
h) пользователь услуг.

3.3 Определения абстрактных синтаксических обозначений

Настоящий стандарт использует следующие термины, определенные спецификацией (см. ИСО/МЭК 8824) абстрактных синтаксических обозначений № 1 (ASN.1):

a) значение;
b) тип;
c) простой тип;
d) структурный тип;
e) тип компонента;
f) тег;
g) теговая разметка;
h) ссылочное имя типа (значения);
i) тип строки символов;
j) булевый тип;
k) true (истина);
l) false (ложь);
m) целый тип;
n) тип битовой строки;
o) тип октетной строки;
p) нулевой тип;
q) тип последовательности;
r) последовательный тип;
s) тегированный тип;
t) тип выбора;
u) тип отбора;
v) действительный тип;
w) тип идентификатора объекта;
x) модуль;
y) разработка;
z) правила кодирования ASN.1;
aa) множество символов ASN.1;
ab) внешний тип.

3.4 Прочие определения

В настоящем стандарте используются следующие термины с соответствующими определениями:

3.4.1 **специализированный в части прикладной ассоциации** (Application Association specific; AA-specific): Прилагательное, используемое для описания объекта с областью применения, которой является прикладная ассоциация (то есть на данное имя можно ссылаться только из прикладной ассоциации, через которую определен объект).

3.4.2 **атрибут** (attribute): Именованная характеристика (параметр) элемента, системы, подсистемы, которая может приобретать конкретное значение на заданном множестве (числа, векторы, символьные выражения, логические значения и т. д.).

3.4.3 **вызываемый MMS-пользователь** (Called MMS-user): MMS-пользователь, выдающий примитив услуги инициирования ответа **Initiate.response**.

3.4.4 **вызывающий MMS-пользователь** (Calling MMS-user): MMS-пользователь, выдающий примитив услуги инициирования запроса **Initiate.request**.

3.4.5 **клиент** (Client): Одноранговая сущность, поддерживающая связь, которая использует VMD с некоторой конкретной целью посредством экземпляра запроса услуги.

3.4.6 **структурный элемент согласованности** (conformance building block; CBB): Элементарный блок, используемый для описания требований согласованности MMS.

3.4.7 **данные** (data): Информация, представленная в формализованном виде, пригодном для передачи, интерпретации или обработки с участием человека или автоматическими средствами.

3.4.8 **домен** (domain): Абстрактный объект, представляющий подмножество возможностей VMD, используемых для особых целей.

3.4.9 **специализированный в части домена (предметно-ориентированный)** (Domain-specific): Прилагательное, используемое для описания объекта, имя которого имеет область применения, являющуюся простым доменом (т. е. ссылаться на данное имя можно из всех прикладных ассоциаций, установленных вместе с VMD, которое может ссылаться на указанную область).

3.4.10 **загрузка** (download): Процесс передачи контента домена, включая любые подчиненные объекты, посредством загрузки данных MMS-пользователю.

3.4.11 **управление событием** (event management): Управление условиями события, действиями события, регистрацией события и перечнем условий события.

3.4.12 **файл** (file): Однозначно поименованный объем информации, имеющий обычный набор атрибутов.

3.4.13 **управление файлом** (file operation): Передача файла между открытыми системами, его инспекция, модификация, замена части контента файла, работа с файлом и его атрибутами.

3.4.14 **файлохранилище** (filestore): Организованное множество файлов, включая их атрибуты и имена, расположенное в конкретной открытой системе.

3.4.15 **информация** (information): Комбинация данных и передаваемого ими смысла.

3.4.16 **некорректный PDU** (invalid PDU): Блок данных PDU, не соответствующий требованиям настоящего стандарта в части структуры и/или содержания.

3.4.17 **журнал** (journal): Множество зарегистрированных событий, изменяемые данные и/или комментарии с временной отметкой, которые могут быть логически упорядочены в процессе получения.

3.4.18 **локальные обстоятельства** (local matter): Решение, принимаемое системой, касающееся ее поведения в спецификации производственных сообщений и не удовлетворяющее требованиям настоящего стандарта.

3.4.19 **механизм протокола производственного сообщения** (Manufacturing Message Protocol Machine; MMPM): Абстрактный механизм выполнения процедур, описанных в настоящем стандарте.

3.4.20 **MMS-контекст** (MMS-context): Спецификация элементов MMS-услуг и семантика связи, используемых в течение срока службы прикладной ассоциации.

3.4.21 **MMS-провайдер** (MMS-provider): Часть сущности приложения, которая концептуально обеспечивает MMS-услуги путем обмена элементов MMS PDU.

3.4.22 **MMS-пользователь** (MMS-user): Часть прикладного процесса, концептуально задействующая спецификацию производственных сообщений.

3.4.23 **отслеживаемое событие** (monitored event): Выявленное изменение состояния условия события.

3.4.24 **событие, запускаемое в сети** (network-triggered event): Запуск, производимый по требованию клиента.

3.4.25 **станция управления** (operator station): Абстрактный объект, представляющий собой оборудование, ассоциированное с VMD и обеспечивающее взаимодействие типа «вход/выход» с оператором.

3.4.26 **предварительно определенный объект** (predefined object): Объект, инстанцированный путем использования механизма, отличного от MMS услуги.

3.4.27 **активизация (вызов) программы** (program Invocation): Абстрактный объект, представляющий собой динамический элемент, наиболее точно соответствующий потоку исполнения в многозадачной среде и составленный из множества доменов.

3.4.28 **ошибка протокола** (protocol error): Блок данных PDU, не соответствующий требованиям настоящего стандарта.

3.4.29 **получающий MMPM** (Receiving MMPM): Механизм MMPM, получающий блоки данных производственных спецификаций MMS PDU.

3.4.30 **получающий MMS-пользователь** (Receiving MMS-user): MMS-пользователь, получающий примитив услуги отображения или подтверждения.

3.4.31 **удаленное устройство управления и мониторинга** (remote device control and monitoring): Изменение или контроль состояния устройства, прикрепленного к ответчику запроса услуги.

3.4.32 **запрашивающий MMS-пользователь** (Requesting MMS-user): MMS-пользователь, выдающий примитив услуги запроса для исполнения.

3.4.33 **ответающийся MMS-пользователь** (Responding MMS-user): MMS-пользователь, выдающий примитив услуги ответа для исполнения.

3.4.34 **семафор** (semaphore): Концептуальный замок, ассоциированный с логическим или физическим ресурсом. Доступ к ресурсу может разрешить только собственник замка.

3.4.35 **управление семафором** (semaphore management): Управление семафором.

3.4.36 **отправляющий MMPM** (Sending MMPM): Механизм MMPM, отправляющий MMS PDU.

3.4.37 **отправляющий MMS-пользователь** (Sending MMS-user): MMS-пользователь, выдающий примитив услуги запроса или ответа.

3.4.38 **сервер** (Server): Одноранговая обобщающая сущность, ведущая себя как VMD агент для конкретного экземпляра запроса услуги.

3.4.39 **стандартизованный объект** (standardized object): Реализация объекта, определение которого приведено в ИСО 9506-1 или в сопутствующем MMS стандарте.

3.4.40 **тип** (type): Абстрактное описание множества значений, выражаемых значением переменой.

3.4.41 **подгрузка** (upload): Процесс передачи контента домена (включая все подчиненные объекты) посредством загрузки данных от удаленного пользователя таким способом, который делает возможным их последующую загрузку.

3.4.42 **корректный PDU** (valid PDU): PDU, удовлетворяющий требованиям настоящего стандарта к структуре и смыслу.

3.4.43 **переменная** (variable): Один или несколько элементов данных, на которые производится ссылка одним именем или описанием.

3.4.44 **доступ к переменной** (variable access): Рассмотрение или модификация переменных (их компонентов), определенных на VMD.

3.4.45 **виртуальное производственное устройство** (Virtual Manufacturing Device; VMD): Абстрактное представление особого множества ресурсов и функциональности действительного производственного устройства, а также отображение указанного абстрактного представления на физические и функциональные аспекты реального производственного устройства.

3.4.46 **удовлетворяющий требованиям VDM (VMD-specific)**: Прилагательное, используемое для описания объекта, имя которого имеет область применения, являющуюся реализацией VMD (т. е. на данное имя могут ссылаться все прикладной ассоциации, установленные вместе с VMD).

4 Сокращения

AA — прикладная ассоциация (application association);
ACSE — сервисный элемент управления ассоциацией (Association Control Service Element);
AE — прикладной логический объект (прикладная сущность) (application entity);
AP — прикладной процесс (application process);
APDU — блок данных прикладного протокола (application protocol data unit);
ASE — прикладной сервисный элемент (application service element);
ASN.1 — абстрактная синтаксическая нотация версия 1 (Abstract Syntax Notation One);
CBB — структурный элемент согласованности (conformance building block);
CIS — утверждение конфигурации и инициализации (Configuration and Initialization Statement);
FRSM — механизм считывания конечного состояния файла (file read state machine);
MMPM — протокольная машина производственного сообщения (Manufacturing Message Protocol Machine);
MMS — спецификация производственных сообщений (Manufacturing Message Specification);
OSI — взаимосвязь открытых систем (Open Systems Interconnection);
PDU — протокольный блок данных (protocol data unit);
ULSM — механизм подкачки конечного состояния (upload state machine);
VMD — виртуальное производственное устройство (Virtual Manufacturing Device).

5 Соглашения

5.1 Соглашения об услугах

Настоящий стандарт основан на положениях, приведенных в соглашениях по определению услуг модели OSI (ИСО/МЭК 10731). Данная модель определяет взаимодействие между MMS-пользователем и MMS-провайдером. Информация передается между MMS-пользователем и MMS-провайдером в виде примитивов услуг с параметрами.

5.2 База числовых значений

Настоящий стандарт использует десятичное представление (систему счисления) всех числовых значений, если не оговорено иное.

5.3 Обозначение

Настоящий стандарт использует абстрактное синтаксическое обозначение, определенное в ИСО/МЭК 8824 (рассматривающем спецификацию ASN.1).

В соответствии с требованиями ASN.1 все ссылки на тип начинаются с большой буквы, все ссылки на значение – с маленькой буквы.

5.4 Поддерживающие разработки

Поддерживающие (вспомогательные) разработки, представленные в различных разделах настоящего стандарта, описаны по месту ссылки, если они использованы преимущественно в одном месте. Если на поддерживающую разработку произведена ссылка много раз из различных мест, то указанные ссылки собраны в конце наиболее содержательного раздела. В любом случае, индекс разработки с указанием номеров страниц может быть размещен в конце настоящего стандарта.

5.5 Сквозные параметры

Многие параметры различных MMS-услуг передаются от примитива запроса (посредством запроса PDU услуги) к примитиву отображения или от примитива ответа (посредством ответа PDU услуги) к примитиву подтверждения без выполнения других действий MMS-провайдером по отношению к рассматриваемому параметру.

5.5.1 Параметры сквозного запроса

Тип, идентифицируемый именем ссылочного типа, должен быть параметром того же имени примитива запроса услуги. Его следует рассматривать как параметр того же имени примитива услуг отображения (при наличии). Значения параметра примитива запроса, примитива отображения и запроса PDU — семантически эквивалентны.

Если это параметр по выбору и он опущен в примитиве запроса услуги, то он должен отсутствовать в запроса PDU. Если параметр по выбору отсутствует в запроса PDU, то он должен отсутствовать в примитиве услуги отображения.

Если параметр присутствует по умолчанию в запроса PDU и указанное значение по умолчанию присутствует в примитиве запроса услуги, то данный параметр может отсутствовать в запроса PDU. Если параметр имеет значение по умолчанию в запроса PDU и данный параметр отсутствует в запроса PDU, то он должен описывать значение по умолчанию в примитиве услуги отображения.

5.5.2 Сквозной параметр ответа

Тип, идентифицируемый именем ссылочного типа, должен быть параметром примитива ответа того же имени. Его следует рассматривать как параметр примитива подтверждения услуги (при наличии) того же имени. Значения параметров примитива ответа, примитива подтверждения и ответа PDU должны быть семантически эквивалентными.

Если это параметр по выбору и он опущен в примитиве ответа услуги, то он должен отсутствовать в ответе PDU. Если параметр по выбору отсутствует в ответе PDU, то он должен отсутствовать в примитиве подтверждения услуги.

Если параметр имеет значение по умолчанию в ответа PDU и данное значение по умолчанию имеется в примитиве ответа услуги, то этот параметр может отсутствовать в ответа PDU. Если параметр имеет значение по умолчанию в PDU ответа и данный параметр отсутствует в ответа PDU, то он должен описывать значение по умолчанию примитива подтверждения услуги.

5.5.3 Перенумерованные значения параметра

Если значения параметров описания услуги перенумерованы, то значения соответствующего параметра протокола должны иметь то же имя (см. 5.5) в примитиве услуги, содержащем указанный параметр. Значения, указанные в примитиве услуги, определенном PDU, и в примитиве услуги, определенном в результате получения примитива услуги, должны быть семантически эквивалентными.

Примечание — Соответствие указанных значений идентифицировано в настоящем стандарте путем использования одинаковых имен в примитивах услуг и в протоколе. В спецификации услуги указанные значения описаны символами верхнего регистра. В спецификации протокола регистр символов имени выбирают в соответствии с синтаксическими требованиями ASN.1. При этом в комментариях после упоминания протокола имя пишут большими буквами.

5.6 Отрицательное подтверждение

Большинство подтвержденных MMS-услуг дают отрицательное подтверждение, если ошибка имеет место при обработке запроса услуги ответающимся MMS-пользователем. Такое отрицательное подтверждение указано параметром **Result(–)** и параметром **ErrorType** (тип ошибки) примитива ответа услуги. Параметр **Result(–)** и параметр **ErrorType** (семантически эквивалентные параметрам примитива ответа) должны быть указаны в примитиве подтверждения услуги.

Абстрактным синтаксисом отрицательного подтверждения является сущность **ErrorPDU** услуги. При этом поле **error** берется из параметра **Problem** примитива ответа услуги.

5.7 Модификатор запроса услуги

MMS-услуги предоставляют возможность использовать модификаторы вместе с экземплярами запросов услуг.

В экземплярах запросов услуг, использующих модификаторы, данные модификаторы, описанные сущностью **RequestPDU,** должны быть семантически эквивалентными (в том же порядке) модификаторам, описанным в примитивах запроса. Примитив отображения должен содержать перечень модификаторов, семантически эквивалентных (и в том же порядке) модификаторам **RequestPDU**.

5.8 Представление данных об ошибках

Для каждой услуги, представленной в настоящем стандарте, ошибки, возникшие вследствие использования указанной услуги, не представлены вместе с протоколом данной услуги. Ошибки описывают в особом разделе.

5.9 Вызывающий и вызванный MMS-пользователи

В настоящем стандарте использованы термины «вызывающий MMS-пользователь» и «вызванный MMS-пользователь». Вызывающий MMS-пользователь — это MMS-пользователь, инициирующий примитив запроса услуг **Initiate.request**. Вызванный MMS-пользователь — это MMS-пользователь, инициирующий примитив ответа услуг **Initiate.response**.

Примечание — Использование термина «вызванный» (**called**) в среде MMS отличается от использования данного термина в среде OSI. В среде MMS термин «вызванный» (**called**) соответствует термину «ответающийся» (**responding**) в среде OSI. Различные термины используют для того, чтобы избежать путаницы при определении запрашивающего/ответающегося MMS-пользователя.

5.10 Отправляющий и получающий MMS-пользователь и MMPM

В настоящем стандарте использованы термины «отправляющий MMS-пользователь» и «получающий MMS-пользователь». Отправляющий MMS-пользователь — это MMS-пользователь, инициирующий примитив услуги запроса или примитив услуги ответа. Получающий MMS-пользователь — это MMS-пользователь, получающий примитив услуги отображения или примитив подтверждения услуги.

Примечание — Важно отметить, что в процессе выполнения подтверждаемой MMS-услуги оба указанных MMS-пользователя являются и отправителями, и получателями в одно и то же время. Первый MMS-пользователь отправляет запрос и получает подтверждение. Тогда как второй MMS-пользователь получает отображение и отправляет ответ.

В настоящем стандарте использованы термины «отправляющий MMPM» и «получающий MMPM». Отправляющий MMPM — это механизм MMPM, отправляющий блок данных производственной спецификации MMS PDU. Получающий MMPM — это механизм MMPM, получающий MMS PDU (расшифровка аббревиатур представлена в разделе 4).

5.11 Запрашивающий и ответающийся MMS-пользователь

В настоящем стандарта использованы термины «запрашивающий MMS-пользователь» и «ответающийся MMS-пользователь». Запрашивающий MMS-пользователь — это MMS-пользователь, инициирующий примитив услуги запроса; ответающийся MMS-пользователь — это MMS-пользователь, инициирующий примитив услуги ответа.

Примечание — Важно отметить, что используемый термин «ответающийся MMS-пользователь» отличается от термина «ответающаяся сущность» в стандарте ACSE и других стандартах. В указанных стандартах данный термин используется для ссылок на сущность, которая ответается на запрос о соединении.

5.12 Клиент и сервер услуг

В настоящем стандарте использованы термины «клиент» и «сервер» для описания модели MMS VMD. Сервер — это одноранговая обобщающая сущность, ведущая себя как VMD для конкретного экземпляра запроса услуги. Клиент — это одноранговая обобщающая сущность, использующая VMD для некоторой конкретной цели посредством экземпляра запроса услуги. Модель VMD преимущественно используется при описании работы сервера и, следовательно, при описании команд и ответов, используемых клиентом. Реальная оконечная система может принять роль клиента, либо роль сервера, либо обе роли в течение срока службы прикладной ассоциации.

5.13 Определения ASN.1

Определения ASN.1, данные в настоящем стандарте (см. разделы 7–23), являются частью модуля ASN.1 «ISO-9506-MMS-1». Определения ASN.1, данные в настоящем стандарте (см. приложение А), являются частью модуля ASN.1 «MMS-Environment-1». Определения ASN.1, данные в настоящем стандарте (приложение В), являются частью модуля ASN.1 «MMS-SCI-Module-1». Определения ASN.1, данные в настоящем стандарте (приложения С, D, и Е), являются частью модуля ASN.1 «ISO-9506-1A». Начальные и конечные утверждения, указывающие, что каждое данное определение ASN.1 является частью соответствующего модуля, опущены для читабельности документа. Каждое определение ASN.1, данное неявно, содержит следующее утверждение

ModuleName DEFINITIONS ::= BEGIN

в начале этого определения. Оно также содержит ключевое слово «**END**» в конце этого определения. Здесь сущность **ModuleName** — это имя модуля ASN.1, для которого рассматриваемое определение является частью.

Примечание — Сущность ISO-9506-MMS-1 указывает на пересмотр № 1 абстрактного синтаксиса ядра MMS, представленного в настоящем стандарте.

5.14 Обозначения подмножества протоколов

Обозначение, представленное в настоящем стандарте, имеет форму языка препроцессора, в который встроено обозначение ASN.1. Это аналогично ситуации в макропрепроцессоре языка С. В рассматриваемой системе символов использованы только три команды:

- IF (<список аргументов>);
- ELSE;
- ENDIF.

Команда **IF** требует указания списка аргументов (в скобках). Аргументы — это структурные элементы согласованности, услуги или параметры. Должны быть указаны один или несколько аргументов. Если имеется более чем один аргумент, то они отделены одним или несколькими пробелами. Аргумент рассмотрен как булева переменная. Она имеет значение **true,** если соответствующая услуга или структурный элемент параметра поддерживается как результат обмена инициированием MMS. Если аргумент только один, то строки, идущие за утверждением **IF** до утверждения **ELSE** (или соответствующего утверждения **ENDIF**, когда утверждение **ELSE** отсутствует), должны быть включены в результирующее определение ASN.1, если поддерживается структурный элемент согласованности с тем же именем. Если имеется более чем один аргумент, то строки, идущие за утверждением **IF**, должны быть включены, если в списке аргументов поддерживается определенный структурный элемент согласованности. Это можно рассматривать как логическую функцию **OR** структурного элемента согласованности.

Утверждения **IF** могут образовывать вложения любой глубины. Смысл функций **IF(*x*)** и **IF(*y*)** заключается в том, чтобы включить строки, следующие за указанными командами, если ***x*** и ***y*** имеют значение **true,** то есть если блок согласованности **x** и блок согласованности **y** (оба блока) включены. Это можно рассматривать как логическую функцию **AND** структурного элемента согласованности.

Утверждение **ELSE** можно использовать, чтобы предоставить возможность включения утверждения ASN.1, если структурный элемент согласованности не имеет значения **true**. Его использование аналогично нормальному использованию утверждения **ELSE** в языках программирования.

Утверждение **ENDIF** используется для указания конца области применения утверждения **IF** или утверждения **ELSE**. Каждое утверждение **IF** должно иметь соответствующее утверждение **ENDIF.**

5.15 Определение эффективного протокола

Протокол, эффективный для любой заданной комбинации услуг и СВВ-параметров, может быть определен с помощью следующей процедуры:

a) для каждой СВВ-услуги и каждого СВВ-параметра, объявленного или оговоренного для обмена типа **Initiate**, задают значение соответствующего аргумента равным **true**;

b) переделывают весь модуль ASN.1, описанный в настоящем стандарте. Для каждого утверждения **IF** оценивают его аргумент:

i) если любой из элементов аргумента имеет значение **true**, то оставляют утверждения, расположенные между утверждением **IF** и соответствующим утверждением **ENDIF** (утверждением **ELSE**, если оно есть). Отменяют утверждения, расположенные между утверждением **ELSE** и соответствующим утверждением **ENDIF**,

ii) если все элементы аргумента имеют значение **false**, то отменяют следующие утверждения до соответствующего утверждения **ELSE** (утверждения **ENDIF**). Если имеется утверждение **ELSE**, то удерживают утверждения, расположенные далее между ним и соответствующим утверждением **ENDIF**,

iii) отменяют утверждение **IF** и соответствующее ему утверждение **ENDIF** (и утверждение **ELSE**, если оно есть). В итоге получается модуль ASN.1, лишенный утверждений **IF, ELSE** и **ENDIF**;

c) в каждой разработке заменяют все комбинации «запятая+правая скобка» на правую скобку;

d) формируют рабочий модуль разработок ASN.1, содержащий только первую разработку (то есть разработку **MMSpdu** раздела 7);

e) добавляют к рабочему модулю ASN.1 любые разработки, на которые производятся ссылки в указанном рабочем модуле и которые в рассматриваемом модуле не содержатся;

f) повторяют шаг e) до тех пор, пока не перестанут добавляться новые разработки.

Результирующий модуль ASN.1 — это модуль, являющийся эффективным для рассматриваемой комбинации СВВ. Подтверждение получения блока данных PDU, не соответствующего настоящему модулю, приведет к выбраковке.

6 Элементы протокольной процедуры

Настоящий раздел содержит описание элементов протокольной процедуры, связанной с отправлением и получением сущностей MMS PDU, а также описание их соотношений с событиями примитивов услуг на стыке MMS-пользователя и MMS-провайдера.

6.1 Описательные соглашения

На рисунках настоящего подраздела использован описательный механизм для стандартной диаграммы состояний. Ниже дано описание данного механизма. Все диаграммы состояний показаны с точки зрения MMS-провайдера.

Каждое состояние представлено прямоугольником. Имя состояния — внутри прямоугольника. Каждая стрелка указывает переход в данное состояние или из данного состояния. Головка стрелки указывает результирующее состояние, как результат перехода.

Каждый переход помечается входным действием, вызывающим переход, и выходными действиями, имеющими место в течение перехода. Входы указаны над выходами. Входы отделены от выходов сплошной горизонтальной линией.

Примитивы услуг с плюсом «+» указывают примитив услуг, содержащий параметр **Result(+)**. Примитивы услуг с минусом «–» указывают примитив услуг, содержащий параметр **Result(–)**.

6.2 Вход и выход из среды MMS

Услуги инициирования, завершения и прерывания доставляют механизмы входа и выхода из среды MMS. Модель указанных услуг (описывающая допустимые последовательности событий) описана в ИСО 9506-1, раздел 8.

6.3 Работа в среде MMS

В среде MMS может быть несколько услуг, одновременно ожидающих выполнения в любой момент времени. ИСО 9506 дает независимое описание диаграммы состояний каждого экземпляра такого запроса услуги.

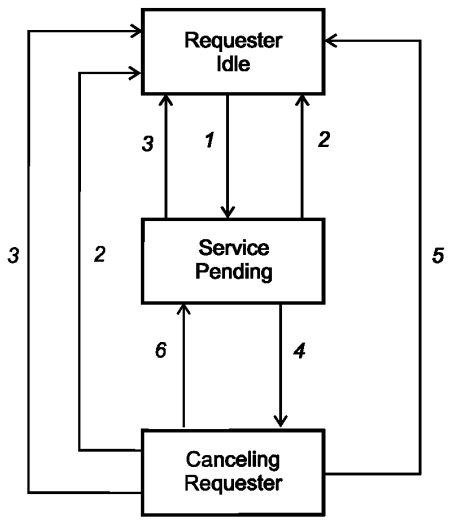
Примечание — В других разделах настоящего стандарта определены дополнительные ограничения на допустимые последовательности примитивов услуг. Они могут дополнительно ограничивать MMS-пользователей.

6.3.1 Подтверждаемые MMS-услуги

Настоящий пункт содержит описание изменений состояния для всех подтверждаемых услуг (услуг с подтверждением), которые могут быть задействованы в среде MMS. Рассматриваемое множество услуг включает все услуги, запрошенные при использовании PDU с пподтверждаемым запросом.

На рисунках 1–2 показаны диаграммы изменения состояния. Они приложимы ко всем указанным услугам и отдельно к каждой реализации каждого запроса услуги. В любой заданный момент времени одновременно могут ожидать выполнения сразу несколько реализаций запросов услуги. Эти реализации подлежат упорядочиванию в соответствии с правилами, установленными в других разделах настоящего стандарта.

Все блоки данных PDU, ассоциированные с выполнением одной реализации подтверждаемой MMS-услуги (рассматриваемые PDU имеют типы **Confirmed-RequestPDU**, **Confirmed-ResponsePDU**, **Confirmed-ErrorPDU**, **Cancel-RequestPDU**, **Cancel-ResponsePDU**, **Cancel-ErrorPDU** и **RejectPDU**) отсылаются в одном и том же контексте представления данных.



Requester Idle	Запрашивающее устройство не активировано
Service Pending	Ожидание услуги
Canceling Requester	Отмена запрашивающего устройства

Переходы:

1 – x.request Confimed-RequestPDU(x)

2 – Confitmed-ResponsePDU(x) x.confirm+

3 – Confimed-ErrorDU(x) x.confirm-

4 – cancel.request Cancel-RequestPDU

5 – Cancel-ResponsePDU and Confirmes-ErrorPDU(x) Cancel.confirm+ and x.confirm-

6 – Cancel-ErrorPDU Cancel.confirm-

Рисунок 1 — Подтверждаемый запрос услуги с точки зрения устройства, запрашивающего услуги

6.3.1.1 Устройство, запрашивающее услуги

Порядок получения блоков информации **Cancel-ResponsePDU** и **Confirmed-ErrorPDU(x)** в переходе 5 (см. 6.3.1) не вполне соответствует установленным требованиям.

В настоящей части стандарта рассмотрено продвижение подтверждаемого запроса MMS-услуги с точки зрения устройства, запрашивающего услуги. Перед выдачей примитива запроса услуги считается, что данная услуга находится в состоянии «запрашивающее устройство не активировано». После получения примитива запроса для любой подтверждаемой MMS-услуги, MMS-провайдер отправляет подтверждение **Confirmed-RequestPDU** (описывая идентификатор задействования, который однозначно идентифицирует реализацию запроса услуги по прикладной ассоциации) и входит в состояние «ожидание услуги».

После получения блока **Confirmed-ResponsePDU** с подтверждаемым ответом, описывающего предварительно запрошенную услугу, и идентификатора вызова, описывающего реализацию данной услуги, MMS-провайдер выдает примитив подтверждения услуги (дающий описание предварительно запрошенного типа услуги и идентификатора вызова) MMS-пользователю, содержащий параметр **Result(+)**. После этого происходит переход в состояние «запрашивающее устройство не активировано».

После получения блока **Confirmed-ErrorPDU**, дающего описание предварительно запрошенной услуги, и идентификатора вызова, содержащего описание реализации услуги, MMS-провайдер выдает примитив подтверждения услуги (дающий описание предварительно запрошенного типа услуги и идентификатора вызова) MMS-пользователю, содержащий параметр **Result(–)**. После этого происходит переход в состояние «запрашивающее устройство не активировано».

После получения примитива услуги отмены запроса от MMS-пользователя, MMS-провайдер отправляет блок **Cancel-RequestPDU**, содержащий идентификатор задействования отменяемого запроса (данная информация содержится в параметрах примитива отмены запроса). При этом система переходит в состояние «отмена запрашивающего устройства».

Состояние «отмена запрашивающего устройства» снимается после подтверждаемого получения одного из четырех возможных входных действий (см. далее).

После получения блока **Cancel-ErrorPDU**, содержащего описание идентификатора активизации, соответствующего рассматриваемой реализации запроса отмены услуги, MMS-провайдер выдает MMS-пользователю примитив услуги отмены подтверждения, содержащий параметр **Result(–),** и возвращается в состояние «ожидание услуги». В данном случае считается, что запрос отмены является неудачным.

Если запрос отмены удачный, то имеют место следующие события:

a) получение блока **Cancel-ResponsePDU**, идентификатор задействования которого соответствует правильной реализации запроса отмены услуги;

b) получение блока **Confirmed-ErrorPDU**, содержащего описание типа отмененной услуги и идентификатор задействования, соответствующий отмененной услуге;

c) MMS-провайдер выдает MMS-пользователю примитив услуги отмены подтверждения, содержащий параметр **Result(+),** и примитив подтверждения отменяемой услуги, содержащий параметр **Result(–)** (поясняющий причину отказа);

d) MMS-провайдер переходит в состояние «запрашивающее устройство неактивно».

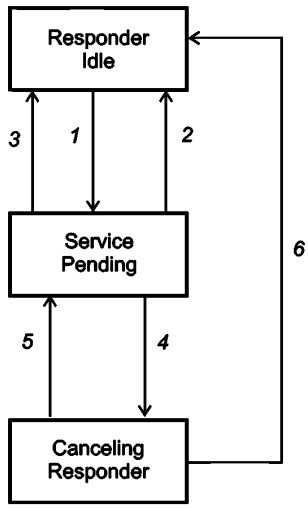
Если получен блок **Confirmed-ResponsePDU**, содержащий описание типа отменяемой услуги, и идентификатор задействования, соответствующий отменяемой услуге, то MMS-провайдер выдает примитив подтверждения услуги, содержащий параметр **Result(+)** услуги, находящейся в процессе отмены. В данном случае запрос отмены считается неудачным, и блок **Cancel-ErrorPDU** будет получен в процессе активизации отмены услуг.

Примечание 1 — Обычно блок **Confirmed-ResponsePDU** подтверждаемого ответа и блок отмены запроса **Cancel-RequestPDU** выдаются одновременно двумя MMS-пользователями в процессе двустороннего диалога.

Если получен блок **Confirmed-ErrorPDU**, содержащий описание типа отменяемой услуги с соответствующим идентификатором активизации, и возникающая ошибка не относится к классу **SERVICE-PREEMPT** и коду **CANCEL**, то MMS-провайдер выдает примитив подтверждения услуги, содержащий параметр **Result(–)** для услуги, находящейся в процессе отмены. В данном случае запрос отмены считается неудачным, и блок **Cancel-ErrorPDU** получается в процессе активизации отмены услуг.

Примечание 2 — Обычно блок **Confirmed-ErrorPDU** подтверждаемой ошибки отменяемой услуги и блок отмены запроса **Cancel-RequestPDU** выдаются одновременно двумя MMS-пользователями в процессе двустороннего диалога.

Обработка ошибочных отмен производится в соответствии с 6.4.



Responder Idle	Ответчик не активирован
Service Pending	Ожидание услуги
Canceling Responder	Отмена активизации ответчика

Переходы:

1 – Confimed-RequestPDU(x)
x.indication

2 – x.response+
Confimed- ResponsePDU(x)

3 – x.response-
Confirmes-ErrorPDU(x)

4 – Cancel-RequestPDU
cancel.indication

5 – Cancel.response-
Cancel.ErrorPDU

6 – Cancel.response + and x.reponse-
Cancel-ResponsePDU and Confirmed-ErrorPDU(x)

Рисунок 2 — Подтверждаемый запрос услуги с точки зрения устройства, предоставляющего услуги

6.3.1.2 Устройство, предоставляющее услуги

Порядок, в котором в переходе 6 на рисунке 2 выдаются примитивы услуг **Cancel.response+** и **x.response-**, может не соответствовать установленным требованиям.

На рисунке 2 показан процесс прохождения подтверждаемого запроса MMS-услуги с точки зрения устройства, предоставляющего услуги. До получения блока данных **Confirmed-RequestPDU** услуги, услуга считается находящейся в состоянии «Ответчик не активирован». После получения блока **Confirmed-RequestPDU** для любой подтверждаемой услуги, идентифицированной выше, MMS-провайдер выдает примитив отображения (дающий спецификацию конкретной запрошенной услуги и идентификатора активизации, содержащего описание реализации услуги) и входит в состояние «ожидание услуги».

После получения примитива услуги ответа, содержащей параметр **Result(+)** (представляет спецификацию ранее указанной услуги и идентификатора активизации, содержащего описание реализации данной услуги), MMS-провайдер отправляет блок данных **Confirmed-ResponsePDU** (представляет спецификацию типа услуги и идентификатора активизации для рассматриваемого примитива ответа). Далее система переходит в состояние «ответчик не активирован».

После получения примитива услуги ответа, содержащей параметр **Result(–)** (представляет спецификацию ранее указанной услуги и идентификатора активизации, содержащего описание реализации данной услуги), MMS-провайдер отправляет блок данных **Confirmed-ErrorPDU** (представляет спецификацию типа услуги и идентификатора активизации из примитива ответа). Далее система переходит в состояние «ответчик не активирован».

После получения блока данных **Cancel-RequestPDU,** представляющего спецификацию идентификатора активизации соответствующей реализации услуги, MMS-провайдер выдает примитив услуги отмены отображения, представляющий спецификацию идентификатора активизации запроса отменяемой услуги (данная информация предоставляется параметром блока **Cancel-RequestPDU**). Далее система переходит в состояние «отмена активизации ответчика».

Примечание 1 — Действия, предпринимаемые при получении блока данных **Cancel-RequestPDU**, идентификатор задействования которого не соответствует ожидающим выполнения реализациям услуги, описаны в 6.4.

Состояние «отмена активизации ответчика» снимается после получения одного из двух возможных входных действий. Они описаны в двух следующих пунктах.

Когда запрос отмены достигает ответающегося MMS-пользователя, имеет место нижеследующая последовательность событий:

a) ответающийся MMS-пользователь выдает ответ отмены, дающий спецификацию идентификатора активизации соответствующей реализации услуги и содержащий параметр **Result(+)**, MMS-провайдеру. Он также выдает примитив услуги ответа, содержащий параметр **Result(–)** (представляет спецификацию класса ошибок **SERVICE-PREEMPT** и кода ошибки **CANCEL**) отменяемой услуги;

b) MMS-провайдер отправляет блоки данных **Cancel-ResponsePDU** и **Confirmed-ErrorPDU**, представляющих спецификацию реализации отменяемой услуги (с классом ошибок **SERVICE-PREEMPT** и кодом ошибки **CANCEL**);

c) MMS-провайдер возвращается в состояние «ответчик не активирован».

MMS-пользователь не должен выдавать примитив услуги отмены ответа, содержащий параметр **Result(+)**, без выдачи примитива услуги ответа, содержащей параметр **Result(–)**, дающий описание класса ошибок **SERVICE-PREEMPT** и кода ошибки **CANCEL**. И наоборот, MMS-пользователь не должен выдавать примитив услуги ответа, содержащей параметр **Result(–)**, представляющий описание класса ошибок **SERVICE-PREEMPT** и кода ошибки **CANCEL**, а также примитив услуги отмены ответа, содержащий параметр **Result(+)**. Таким образом, указанные два события логически происходят вместе.

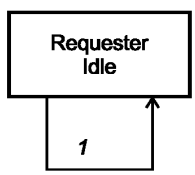
Если получен ответ отмены, дающий спецификацию идентификатора активизации соответствующей реализации услуги, содержащего параметр **Result(–)**, то MMS-провайдер отправляет блок данных **Cancel-ErrorPDU** и возвращается к состоянию «ожидание услуги». В данном случае запрос отмены считается неудачным.

Примечание 2 — Обработка ошибочных запросов отмены и некорректных блоков данных PDU описана в 6.4.

6.3.2 Неподтверждаемые MMS-услуги

Настоящий пункт содержит описание порядка выполнения неподтвержденных MMS-услуг. Данное множество услуг определено как услуги, совершающие выбор **UnconfirmedService**, определенный в разделе 7.

На рисунках 3–4 даны диаграммы изменения состояния, приложенные к каждой указанной выше услуге и используемые отдельно для каждой реализации каждого запроса услуги.



Requester Idle	Устройство, запрашивающее услуги, не активировано

Переход:
1 –v.request
Uniconfirmed-PDU(y)

Рисунок 3 — Неподтвержденная услуга с точки зрения устройства, запрашивающего услуги

6.3.2.1 Устройство, запрашивающее услуги

На рисунке 3 показан процесс прохождения неподтверждаемой MMS-услуги с точки зрения устройства, запрашивающего эти услуги. Перед выдачей примитива запроса услуги считается, что услуга находится в состоянии «запрашивающее устройство не активировано». После получения примитива запроса для любой из вышеуказанных неподтверждаемых услуг, MMS-провайдер отправляет блок данных **Unconfirmed-PDU** (представляющий спецификацию конкретной запрашиваемой услуги) и переходит назад в состояние «запрашиваемое устройство не активировано».

Для неподтвержденных MMS-услуг ответа PDU или ошибки PDU не доставляются. Далее показано, что отменить неподтвержденную MMS-услугу невозможно.

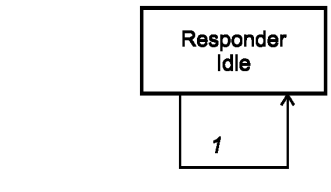


Рисунок 4 — Неподтвержденная услуга с точки зрения устройства, предоставляющего услуги

6.3.2.2 Устройство, предоставляющее услуги

На рисунке 4 показан процесс прохождения неподтверждаемой MMS-услуги с точки зрения устройства, предоставляющего эти услуги. Перед получением блока данных **Unconfirmed-PDU** считается, что услуга находится в состоянии «ответчик не активирован». После получения примитива запроса для любой вышеуказанной неподтверждаемой услуги, MMS-провайдер выдает примитив услуги отображения (представляющий спецификацию конкретной запрашиваемой услуги, основанной на информации полученного блока **Unconfirmed-PDU)** и переходит из состояния «ответчик не активирован» в состояние «ответчик не активирован» (в то же самое состояние!).

Для неподтвержденных MMS-услуг, MMS-пользователь может не выдавать какого-либо примитива ответа. Ниже показано, что отменить неподтвержденную MMS-услугу невозможно.

6.3.3 Услуга отмены

Услуга отмены (в этом случае, если это подтвержденная услуга) работает не так, как другие подтверждаемые услуги. Если услуга отмены задействована, то механизм перехода в нужное состояние не работает. И действительно, здесь оказывается поврежденным механизм перехода отменяемой услуги в требуемое состояние. Запрос услуги отмены не может быть отменен повторным задействованием услуги отмены. Идентификатор задействования, описанный в запросе услуги отмены, не может быть идентификатором повторного задействования услуги отмены, так как данная услуга работает только с теми услугами, где сделан запрос подтверждаемой услуги **ConfirmedServiceRequest** (определение приведено в разделе 7).

В любой заданный момент времени и для любого заданного экземпляра запроса услуги в состоянии ожидания выполнения может находиться только одна услуга отмены. Задействование услуги отмены не оказывает влияния на число запросов услуги, ожидающей выполнения в любой заданный момент времени согласно услуге запуска. Запросы услуги отмены не учитываются при определении факта достижения установленного предела.

Особенности работы услуги отмены описаны в предшествующих разделах настоящего документа: это работа устройства, запрашивающего услуги, и устройства, предоставляющего услуги для подтвержденных MMS-услуг.

6.4 Обработка условий ошибок

После получения некорректного блока данных PDU, MMS-провайдер выдает примитив услуги выбраковки отображения MMS-пользователю, идентифицируя выявленную ошибку, а также отправляет блок **RejectPDU** в систему, из которой был получен некорректный PDU.

В данном случае не должно происходить изменения состояния. Если некорректный блок данных PDU является некорректным блоком **RejectPDU**, то **RejectPDU** не отсылается.

После получения блока отмены запроса **Cancel-RequestPDU**, в котором делается попытка отменить запрос неизвестной услуги (например, когда описанный идентификатор задействования не относится к ожидающей выполнения подтверждаемой услуге), MMS-провайдер отправляет блок данных **Cancel-ErrorPDU** отправителю запроса отмены. В данном случае MMS-пользователь не уведомляется об ошибочной попытке отмены.

Примечание 1 — Возможно возникновение ситуации, при которой блоки **Cancel-RequestPDU, Confirmed-ResponsePDU** или **Confirmed-ErrorPDU** рассматриваемой отменяемой услуги выдаются одновременно двумя общающимися MMS-пользователями. Тогда одна сторона считает, что услуга выполнена, а другая сторона считает, что услуга ожидает отмены. В данном случае запрос отмены срывается, и услуга выполняется в нормальном режиме.

После получения блока данных **Cancel-ErrorPDU**, где механизм смены состояний (на который производится ссылка идентификатором задействования отменяемой услуги) находится в состоянии «запрашиваемое устройство не активировано», MMS-провайдер выдает примитив услуги отмены подтверждения MMS-пользователю.

Примечание 2 — Данный случай имеет место, когда блок данных **Confirmed-ResponsePDU** (блок данных **Confirmed-ErrorPDU)** отменяемой услуги считает корректным блок **Cancel-RequestPDU** данной услуги.

6.5 Услуга выбраковки и блок данных RejectPDU

Услугу выбраковки используют для уведомления MMS-пользователя об имеющихся ошибках протокола. Работа настоящей услуги описана в ИСО 9506-1, раздел 7.

Примечание — Действия, предпринимаемые MMS-пользователем после получения примитива услуги выбраковки отображения, имеют локальный характер. Важно отметить, что вследствие возможности выбраковки запроса, ответа или ошибки, два общающихся MMS-пользователя могут по-разному понимать состояние объектов, ожидающих транзакции (см. ИСО 9506-1, раздел 7). Услуга прерывания может использоваться в любой момент MMS-пользователем для завершения пребывания в среде **MMS-Environment** и завершения прикладной ассоциации.

7 Блоки данных протокола спецификации производственных сообщений MMS PDU

Настоящий раздел описывает блоки данных PDU, используемые для обработки протокола MMS. Отображение указанных PDU на услуги нижнего уровня описано в разделе 24. Отображение MMS-услуг на указанные блоки данных PDU описано в разделах 8–23.

```
ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) }
DEFINITIONS ::= BEGIN
        EXPORTS AlternateAccess,
        AttachToEventCondition,
        AttachToSemaphore,
        ConfirmedServiceRequest,
        Data,
        EE-State,
        FileName,
        Identifier,
        Integer8,
        Integer32,
        MMSString,
        MMS255String,
        ObjectName,
        TimeOfDay,
        TypeSpecification,
        Unsigned32,
        Unsigned8,
        VariableSpecification;
        IMPORTS ApplicationReference,
        Authentication-value FROM
        MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1 (4) }
        ObtainFile-Request,
        ObtainFile-Response,
        ObtainFile-Error,
        FileOpen-Request,
        FileOpen-Response,
    FileRead-Request,
        FileRead-Response,
        FileClose-Request,
        FileClose-Response,
```

```
                FileRename-Request,
                FileRename-Response,
                FileRename-Error,
                FileDelete-Request,
                FileDelete-Response,
                FileDirectory-Request,
                FileDirectory-Response,
                DefineScatteredAccess-Request,
                DefineScatteredAccess-Response,
                ScatteredAccessDescription,
                GetScatteredAccessAttributes-Request,
                GetScatteredAccessAttributes-Response FROM
                ISO-9506-MMS-1A { iso standard 9506 part(2) mms-annex-version1(3) }
                AccessCondition,
                AdditionalCBBOptions,
                AdditionalSupportOptions,
                Address,
                AlarmAckRule,
                Control-State,
                DomainState,
                EC-State,
                EC-Class,
                EE-Duration,
                EE-Class,
                EventTime,
                Journal-Variable,
                LogicalStatus,
                Modifier,
                normalPriority,
                normalSeverity,
                ParameterSupportOptions,
                PhysicalStatus,
                Priority,
                ProgramInvocationState,
                Running-Mode,
                ServiceSupportOptions,
                Severity,
                Transitions,
                TypeDescription,
                ULState,
                VMDState
                FROM MMS-Object-Module-1
                { iso standard 9506 part(1) mms-object-model-version1(2) };
MMSpdu ::= CHOICE {
                confirmed-RequestPDU                    [0] IMPLICIT Confirmed-RequestPDU,
                confirmed-ResponsePDU                   [1] IMPLICIT Confirmed-ResponsePDU,
                confirmed-ErrorPDU                      [2] IMPLICIT Confirmed-ErrorPDU,
IF ( unsolicitedStatus informationReport eventNotification )
                unconfirmed-PDU                         [3] IMPLICIT Unconfirmed-PDU,
ELSE
                unconfirmed-PDU                         [3] IMPLICIT NULL,
ENDIF
                rejectPDU                               [4] IMPLICIT RejectPDU,
IF (cancel)
                cancel-RequestPDU                       [5] IMPLICIT Cancel-RequestPDU,
```

```
        cancel-ResponsePDU                  [6] IMPLICIT Cancel-ResponsePDU,
        cancel-ErrorPDU                     [7] IMPLICIT Cancel-ErrorPDU,
ELSE
        cancel-RequestPDU                   [5] IMPLICIT NULL,
        cancel-ResponsePDU                  [6] IMPLICIT NULL,
        cancel-ErrorPDU                     [7] IMPLICIT NULL,
ENDIF
        initiate-RequestPDU                 [8] IMPLICIT Initiate-RequestPDU,
        initiate-ResponsePDU                [9] IMPLICIT Initiate-ResponsePDU,
        initiate-ErrorPDU                   [10] IMPLICIT Initiate-ErrorPDU,
        conclude-RequestPDU                 [11] IMPLICIT Conclude-RequestPDU,
        conclude-ResponsePDU                [12] IMPLICIT Conclude-ResponsePDU,
        conclude-ErrorPDU                   [13] IMPLICIT Conclude-ErrorPDU
        }
```

Существуют 14 типов блоков данных протоколов PDU в MMS. В разделе 8 определены блоки **Initiate-RequestPDU**, **Initiate-ResponsePDU**, **Initiate-ErrorPDU**, **Conclude-RequestPDU**, **Conclude-ResponsePDU**, **Conclude-ErrorPDU**, **RejectPDU**, **Cancel-RequestPDU**, **Cancel-ResponsePDU** и **Cancel-ErrorPDU**. Оставшиеся типы PDU определены в 7.1–7.4.

7.1 Confirmed-RequestPDU (подтверждение запроса)

```
   Confirmed-RequestPDU ::= SEQUENCE {
        invokeID                            Unsigned32,
IF (attachToEventCondition attachToSemaphore )
        listOfModifiers                 SEQUENCE OF Modifier OPTIONAL,
ENDIF
        service                             ConfirmedServiceRequest,
        ...
IF ( csr cspi ),
        service-ext                         [79] Request-Detail OPTIONAL
ENDIF
                                -- shall not be transmitted if value is the value
                                -- of a tagged type derived from NULL
        }
```

Confirmed-RequestPDU — это последовательность, содержащая четыре элемента: целое без знака, перечень модификаторов по выбору, подтверждаемый запрос услуги **ConfirmedServiceRequest** и подробности запроса **Request-Detail**.

Идентификатор задействования **InvokeID —** это 32-битное целое без знака. Он однозначно идентифицирует запрос услуги среди всех ожидающих выполнения подтвержденных запросов услуги от конкретного MMS-пользователя по заданной прикладной ассоциации. В любой момент времени должен иметь место самое большее один ожидающий выполнения запрос услуги от конкретного MMS-пользователя по некоторой прикладной ассоциации для любого заданного идентификатора задействования **InvokeID**. Значение **InvokeID** указано MMS-пользователем в примитиве запроса услуг (см. ИСО 9506-1, раздел 5). Значение **InvokeID**, указанное в **Confirmed-ResponsePDU** и **Confirmed-ErrorPDU,** предоставляет возможность MMS-провайдеру и MMS-пользователю коррелировать указанные PDU с рассматриваемым запросом услуги.

Перечень модификаторов **listOfModifiers** нужен для назначения модификаторов при выполнении запросов услуг. Модификатор, описанный в перечне **listOfModifiers**, должен быть успешно выполнен до начала выполнения следующего модификатора перечня **listOfModifiers** или до начала выполнения запроса услуги. Поэтому порядок следования модификаторов в перечене очень важен. Если перечня **listOfModifiers** нет, то выполнение запроса услуги может начаться сразу после получения запроса без предварительных условий.

Запрос подтверждаемой услуги **ConfirmedServiceRequest** нужен для ее идентификации и аргумента. Данный параметр описан в 7.1.1.

Действие модификатора моделируется механизмом смены состояний задействования услуги, рассмотренным в разделе 6. Определения модификаторов содержатся в описаниях протокола в следующих разделах настоящего стандарта.

7.1.1 ConfirmedServiceRequest (запрос подтверждаемой услуги)

```
    ConfirmedServiceRequest ::= CHOICE {
IF ( status )
        status
            [0] IMPLICIT Status-Request
ELSE
        status
            [0] IMPLICIT NULL
ENDIF
IF ( getNameList )
,       getNameList
            [1] IMPLICIT GetNameList-Request
ELSE
,       getNameList
            [1] IMPLICIT NULL
ENDIF
IF ( identify )
,       identify
            [2] IMPLICIT Identify-Request
ELSE
,       identify
            [2] IMPLICIT NULL
ENDIF
IF ( rename )
,       rename
            [3] IMPLICIT Rename-Request
ELSE
,       rename
            [3] IMPLICIT NULL
ENDIF
IF ( read )
,       read
            [4] IMPLICIT Read-Request
ELSE
,       read
            [4] IMPLICIT NULL
ENDIF
IF ( write )
,       write
            [5] IMPLICIT Write-Request
ELSE
,       write
            [5] IMPLICIT NULL
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
,       getVariableAccessAttributes
            [6] GetVariableAccessAttributes-Request
ELSE
,       getVariableAccessAttributes
            [6] IMPLICIT NULL
ENDIF
ELSE
,       getVariableAccessAttributes
            [6] IMPLICIT NULL
```

```
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
,       defineNamedVariable
            [7] IMPLICIT DefineNamedVariable-Request
ELSE
,       defineNamedVariable
            [7] IMPLICIT NULL
ENDIF
ELSE
,       defineNamedVariable
            [7] IMPLICIT NULL
ENDIF
ELSE
,       defineNamedVariable
            [7] IMPLICIT NULL
ENDIF
IF ( vsca )
        -- [8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
,       defineScatteredAccess
            [8] IMPLICIT DefineScatteredAccess-Request
ELSE
,       defineScatteredAccess
            [8] IMPLICIT NULL
ENDIF
        -- [9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
,       getScatteredAccessAttributes
            [9] GetScatteredAccessAttributes-Request
ELSE
,       getScatteredAccessAttributes
            [9] IMPLICIT NULL
ENDIF
ELSE
,       defineScatteredAccess
            [8] IMPLICIT NULL,
        getScatteredAccessAttributes
            [9] IMPLICIT NULL
ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
,       deleteVariableAccess
            [10] IMPLICIT DeleteVariableAccess-Request
ELSE
,       deleteVariableAccess
            [10] IMPLICIT NULL
ENDIF
ELSE
,       deleteVariableAccess
            [10] IMPLICIT NULL
ENDIF
IF ( vlis )
IF ( vnam )
```

```
IF ( defineNamedVariableList )
,       defineNamedVariableList
            [11] IMPLICIT DefineNamedVariableList-Request
ELSE
,       defineNamedVariableList
            [11] IMPLICIT NULL
ENDIF
IF ( getNamedVariableListAttributes )
,       getNamedVariableListAttributes
            [12] GetNamedVariableListAttributes-Request
ELSE
,       getNamedVariableListAttributes
            [12] IMPLICIT NULL
ENDIF
IF ( deleteNamedVariableList )
,       deleteNamedVariableList
            [13] IMPLICIT DeleteNamedVariableList-Request
ELSE
,       deleteNamedVariableList
            [13] IMPLICIT NULL
ENDIF
ELSE
,       defineNamedVariableList
            [11] IMPLICIT NULL,
        getNamedVariableListAttribute
            [12] IMPLICIT NULL,
        deleteNamedVariableList
            [13] IMPLICIT NULL
ENDIF
ELSE
,       defineNamedVariableList
            [11] IMPLICIT NULL,
        getNamedVariableListAttributes
            [12] IMPLICIT NULL,
        deleteNamedVariableList
            [13] IMPLICIT NULL
ENDIF
ENDIF
IF ( vnam )
IF ( defineNamedType )
,       defineNamedType
            [14] IMPLICIT DefineNamedType-Request
ELSE
,       defineNamedType
            [14] IMPLICIT NULL
ENDIF
IF ( getNamedTypeAttributes )
,       getNamedTypeAttributes
            [15] GetNamedTypeAttributes-Request
ELSE
,       getNamedTypeAttributes
            [15] IMPLICIT NULL
ENDIF
IF ( deleteNamedType )
,       deleteNamedType
```

```
            [16] IMPLICIT DeleteNamedType-Request
ELSE
,           deleteNamedType
            [16] IMPLICIT NULL
ENDIF
ELSE
,           defineNamedType
            [14] IMPLICIT NULL,
            getNamedTypeAttributes
            [15] IMPLICIT NULL,
            deleteNamedType
            [16] IMPLICIT NULL
ENDIF
IF ( input )
,           input
            [17] IMPLICIT Input-Request
ELSE
,           input
            [17] IMPLICIT NULL
ENDIF
IF ( output )
,           output
            [18] IMPLICIT Output-Request
ELSE
,           output
            [18] IMPLICIT NULL
ENDIF
IF ( takeControl )
,           takeControl
            [19] IMPLICIT TakeControl-Request
ELSE
,           takeControl
            [19] IMPLICIT NULL
ENDIF
IF ( relinquishControl )
,           relinquishControl
            [20] IMPLICIT RelinquishControl-Request
ELSE
,           relinquishControl
            [20] IMPLICIT NULL
ENDIF
IF ( defineSemaphore )
,           defineSemaphore
            [21] IMPLICIT DefineSemaphore-Request
ELSE
,         defineSemaphore
            [21] IMPLICIT NULL
ENDIF
IF ( deleteSemaphore )
,           deleteSemaphore
            [22] DeleteSemaphore-Request
ELSE
,           deleteSemaphore
            [22] IMPLICIT NULL
ENDIF
```

```
IF ( reportSemaphoreStatus )
,       reportSemaphoreStatus
            [23] ReportSemaphoreStatus-Request
ELSE
,       reportSemaphoreStatus
            [23] IMPLICIT NULL
ENDIF
IF ( reportPoolSemaphoreStatus )
,       reportPoolSemaphoreStatus
            [24] IMPLICIT ReportPoolSemaphoreStatus-Request
ELSE
,       reportPoolSemaphoreStatus
            [24] IMPLICIT NULL
ENDIF
IF ( reportSemaphoreEntryStatus )
,       reportSemaphoreEntryStatus
            [25] IMPLICIT ReportSemaphoreEntryStatus-Request
ELSE
,       reportSemaphoreEntryStatus
            [25] IMPLICIT NULL
ENDIF
IF ( initiateDownloadSequence )
,       initiateDownloadSequence
            [26] IMPLICIT InitiateDownloadSequence-Request,
        downloadSegment
            [27] IMPLICIT DownloadSegment-Request,
        terminateDownloadSequence
            [28] IMPLICIT TerminateDownloadSequence-Request
ELSE
,       initiateDownloadSequence
            [26] IMPLICIT NULL,
        downloadSegment
            [27] IMPLICIT NULL,
        terminateDownloadSequence
            [28] IMPLICIT NULL
ENDIF
IF ( initiateUploadSequence )
,       initiateUploadSequence
            [29] IMPLICIT InitiateUploadSequence-Request,
        uploadSegment
            [30] IMPLICIT UploadSegment-Request,
        terminateUploadSequence
            [31] IMPLICIT TerminateUploadSequence-Request
ELSE
,       initiateUploadSequence
            [29] IMPLICIT NULL,
        uploadSegment
            [30] IMPLICIT NULL,
        terminateUploadSequence
            [31] IMPLICIT NULL
ENDIF
IF ( requestDomainDownload )
,       requestDomainDownload
            [32] IMPLICIT RequestDomainDownload-Request
ELSE
```

```
,           requestDomainDownload
                [32] IMPLICIT NULL
ENDIF
IF ( requestDomainUpload )
,           requestDomainUpload
                [33] IMPLICIT RequestDomainUpload-Request
ELSE
,           requestDomainUpload
                [33] IMPLICIT NULL
ENDIF
IF ( loadDomainContent )
,           loadDomainContent
                [34] IMPLICIT LoadDomainContent-Request
ELSE
,           loadDomainContent
                [34] IMPLICIT NULL
ENDIF
IF ( storeDomainContent )
,       storeDomainContent
                [35] IMPLICIT StoreDomainContent-Request
ELSE
,           storeDomainContent
                [35] IMPLICIT NULL
ENDIF
IF ( deleteDomain )
,           deleteDomain
                [36] IMPLICIT DeleteDomain-Request
ELSE
,           deleteDomain
                [36] IMPLICIT NULL
ENDIF
IF ( getDomainAttributes )
,           getDomainAttributes
                [37] IMPLICIT GetDomainAttributes-Request
ELSE
,           getDomainAttributes
                [37] IMPLICIT NULL
ENDIF
IF ( createProgramInvocation )
,           createProgramInvocation
                [38] IMPLICIT CreateProgramInvocation-Request
ELSE
,           createProgramInvocation
                [38] IMPLICIT NULL
ENDIF
IF ( deleteProgramInvocation )
,           deleteProgramInvocation
                [39] IMPLICIT DeleteProgramInvocation-Request
ELSE
,           deleteProgramInvocation
                [39] IMPLICIT NULL
ENDIF
IF ( start )
,           start
                [40] IMPLICIT Start-Request
```

```
ELSE
,       start
            [40] IMPLICIT NULL
ENDIF
IF ( stop )
,       stop
            [41] IMPLICIT Stop-Request
ELSE
,       stop
            [41] IMPLICIT NULL
ENDIF
IF ( resume )
,       resume
            [42] IMPLICIT Resume-Request
ELSE
,       resume
            [42] IMPLICIT NULL
ENDIF
IF ( reset )
,       reset
            [43] IMPLICIT Reset-Request
ELSE
,       reset
            [43] IMPLICIT NULL
ENDIF
IF ( kill )
,       kill
            [44] IMPLICIT Kill-Request
ELSE
,       kill
            [44] IMPLICIT NULL
ENDIF
IF ( getProgramInvocationAttributes )
,       getProgramInvocationAttributes
            [45] IMPLICIT GetProgramInvocationAttributes-Request
ELSE
,       getProgramInvocationAttributes
            [45] IMPLICIT NULL
ENDIF
IF ( obtainFile )
,       obtainFile
            [46] IMPLICIT ObtainFile-Request
ELSE
,       obtainFile
            [46] IMPLICIT NULL
ENDIF
IF ( defineEventCondition )
,       defineEventCondition
            [47] IMPLICIT DefineEventCondition-Request
ELSE
,       defineEventCondition
            [47] IMPLICIT NULL
ENDIF
IF ( deleteEventCondition )
,       deleteEventCondition
```

```
                [48] DeleteEventCondition-Request
ELSE
,           deleteEventCondition
                [48] IMPLICIT NULL
ENDIF
IF ( getEventConditionAttributes )
,           getEventConditionAttributes
                [49] GetEventConditionAttributes-Request
ELSE
,           getEventConditionAttributes
                [49] IMPLICIT NULL
ENDIF
IF ( reportEventConditionStatus )
,           reportEventConditionStatus
                [50] ReportEventConditionStatus-Request
ELSE
,           reportEventConditionStatus
                [50] IMPLICIT NULL
ENDIF
IF ( alterEventConditionMonitoring )
,           alterEventConditionMonitoring
                [51] IMPLICIT AlterEventConditionMonitoring-Request
ELSE
,           alterEventConditionMonitoring
                [51] IMPLICIT NULL
ENDIF
IF ( triggerEvent )
,           triggerEvent
                [52] IMPLICIT TriggerEvent-Request
ELSE
,           triggerEvent
                [52] IMPLICIT NULL
ENDIF
IF ( defineEventAction )
,           defineEventAction
                [53] IMPLICIT DefineEventAction-Request
ELSE
,           defineEventAction
                [53] IMPLICIT NULL
ENDIF
IF ( deleteEventAction )
,           deleteEventAction
                [54] DeleteEventAction-Request
ELSE
,           deleteEventAction
                [54] IMPLICIT NULL
ENDIF
IF ( getEventActionAttributes )
,           getEventActionAttributes
                [55] GetEventActionAttributes-Request
ELSE
,           getEventActionAttributes
                [55] IMPLICIT NULL
ENDIF
IF ( reportEventActionStatus )
```

```
,       reportEventActionStatus
            [56] ReportEventActionStatus-Request
ELSE
,       reportEventActionStatus
            [56] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
,       defineEventEnrollment
            [57] IMPLICIT DefineEventEnrollment-Request
ELSE
,       defineEventEnrollment
            [57] IMPLICIT NULL
ENDIF
IF ( deleteEventEnrollment )
,       deleteEventEnrollment
            [58] DeleteEventEnrollment-Request
ELSE
,       deleteEventEnrollment
            [58] IMPLICIT NULL
ENDIF
IF ( alterEventEnrollment )
,       alterEventEnrollment
            [59] IMPLICIT AlterEventEnrollment-Request
ELSE
,       alterEventEnrollment
            [59] IMPLICIT NULL
ENDIF
IF ( reportEventEnrollmentStatus )
,       reportEventEnrollmentStatus
            [60] ReportEventEnrollmentStatus-Request
ELSE
,       reportEventEnrollmentStatus
            [60] IMPLICIT NULL
ENDIF
IF ( getEventEnrollmentAttributes )
,       getEventEnrollmentAttributes
            [61] IMPLICIT GetEventEnrollmentAttributes-Request
ELSE
,       getEventEnrollmentAttributes
            [61] IMPLICIT NULL
ENDIF
IF ( acknowledgeEventNotification )
,       acknowledgeEventNotification
            [62] IMPLICIT AcknowledgeEventNotification-Request
ELSE
,       acknowledgeEventNotification
            [62] IMPLICIT NULL
ENDIF
IF ( getAlarmSummary )
,       getAlarmSummary
            [63] IMPLICIT GetAlarmSummary-Request
ELSE
,       getAlarmSummary
            [63] IMPLICIT NULL
ENDIF
```

```
IF ( getAlarmEnrollmentSummary )
,       getAlarmEnrollmentSummary
            [64] IMPLICIT GetAlarmEnrollmentSummary-Request
ELSE
,       getAlarmEnrollmentSummary
            [64] IMPLICIT NULL
ENDIF
IF ( readJournal )
,       readJournal
            [65] IMPLICIT ReadJournal-Request
ELSE
,       readJournal
            [65] IMPLICIT NULL
ENDIF
IF ( writeJournal )
,       writeJournal
            [66] IMPLICIT WriteJournal-Request
ELSE
,       writeJournal
            [66] IMPLICIT NULL
ENDIF
IF ( initializeJournal )
,       initializeJournal
            [67] IMPLICIT InitializeJournal-Request
ELSE
,       initializeJournal
            [67] IMPLICIT NULL
ENDIF
IF ( reportJournalStatus )
,       reportJournalStatus
            [68] ReportJournalStatus-Request
ELSE
,       reportJournalStatus
            [68] IMPLICIT NULL
ENDIF
IF ( createJournal )
,       createJournal
            [69] IMPLICIT CreateJournal-Request
ELSE
,       createJournal
            [69] IMPLICIT NULL
ENDIF
IF ( deleteJournal )
,       deleteJournal
            [70] IMPLICIT DeleteJournal-Request
ELSE
,       deleteJournal
            [70] IMPLICIT NULL
ENDIF
IF ( getCapabilityList )
,       getCapabilityList
            [71] IMPLICIT GetCapabilityList-Request
ELSE
,       getCapabilityList
            [71] IMPLICIT NULL
```

```
ENDIF
        -- choices [72] through [77] are reserved for use by services
        -- defined in annex D
IF ( fileOpen )
,       fileOpen
            [72] IMPLICIT FileOpen-Request
ELSE
,       fileOpen
            [72] IMPLICIT NULL
ENDIF
IF ( fileRead )
,       fileRead
            [73] IMPLICIT FileRead-Request
ELSE
,       fileRead
            [73] IMPLICIT NULL
ENDIF
IF ( fileClose )
,       fileClose
            [74] IMPLICIT FileClose-Request
ELSE
,       fileClose
            [74] IMPLICIT NULL
ENDIF
IF ( fileRename )
,       fileRename
            [75] IMPLICIT FileRename-Request
ELSE
,       fileRename
            [75] IMPLICIT NULL
ENDIF
IF ( fileDelete )
,       fileDelete
            [76] IMPLICIT FileDelete-Request
ELSE
,       fileDelete
            [76] IMPLICIT NULL
ENDIF
IF ( fileDirectory )
,       fileDirectory
            [77] IMPLICIT FileDirectory-Request
ELSE
,       fileDirectory
            [77] IMPLICIT NULL
ENDIF
,...
IF ( csr cspi )
,       additionalService
            [78] AdditionalService-Request
ENDIF
        -- choice [79] is reserved
IF ( getDataExchangeAttributes )
,       getDataExchangeAttributes
            [80] GetDataExchangeAttributes-Request
                        -- Shall not appear in minor version 1
```

```
ENDIF
IF ( exchangeData )
,       exchangeData
            [81] IMPLICIT ExchangeData-Request
                        -- Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList )
,       defineAccessControlList
            [82] IMPLICIT DefineAccessControlList-Request
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes )
,       getAccessControlListAttributes
            [83] GetAccessControlListAttributes-Request
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( reportAccessControlledObjects )
,       reportAccessControlledObjects
            [84] IMPLICIT ReportAccessControlledObjects-Request
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( deleteAccessControlList )
,       deleteAccessControlList
            [85] IMPLICIT DeleteAccessControlList-Request
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl )
,       changeAccessControl
            [86] IMPLICIT ChangeAccessControl-Request
                        -- Shall not appear in minor version 1 or 2
ENDIF
        ,...
        }
```

Тип запроса **ConfirmedServiceRequest** должен идентифицировать тип услуги и аргумент данной услуги. Теги контекста идентифицируют тип услуги. Определения каждой индивидуальной услуги описывают форму аргумента данной услуги путем определения типа, на который произведена ссылка при разработке запроса **ConfirmedServiceRequest**. Каждая услуга сущности **ConfirmedServiceRequest** является подтверждаемой.

7.1.2 AdditionalService-Request (запрос дополнительной услуги)

```
AdditionalService-Request ::= CHOICE {
IF ( csr )
IF ( vMDStop )
        vMDStop
                [0] IMPLICIT VMDStop-Request
ELSE
        vMDStop
                [0] IMPLICIT NULL
ENDIF
IF ( vMDReset )
,       vMDReset
                [1] IMPLICIT VMDReset-Request
ELSE
,       vMDReset
                [1] IMPLICIT NULL
```

```
ENDIF
IF ( select )
,       select
                [2] IMPLICIT Select-Request
ELSE
,       select
                [2] IMPLICIT NULL
ENDIF
IF ( alterProgramInvocationAttributes )
,       alterPI
                [3] IMPLICIT AlterProgramInvocationAttributes-Request
ELSE
,       alterPI
                [3] IMPLICIT NULL
ENDIF
ELSE
,       vMDStop
                [0] IMPLICIT NULL,
        vMDReset
                [1] IMPLICIT NULL,
        select
                [2] IMPLICIT NULL,
        alterPI
                [3] IMPLICIT NULL
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
,       initiateUCLoad
                [4] IMPLICIT InitiateUnitControlLoad-Request
ELSE
,       initiateUCLoad
                [4] IMPLICIT NULL
ENDIF
IF ( unitControlLoadSegment )
,       uCLoad
                [5] IMPLICIT UnitControlLoadSegment-Request
ELSE
,       uCLoad
                [5] IMPLICIT NULL
ENDIF
IF ( unitControlUpload )
,       uCUpload
                [6] IMPLICIT UnitControlUpload-Request
ELSE
,       uCUpload
                [6] IMPLICIT NULL
ENDIF
IF ( startUnitControl )
,       startUC
                [7] IMPLICIT StartUnitControl-Request
ELSE
,       startUC
                [7] IMPLICIT NULL
ENDIF
IF ( stopUnitControl )
```

```
,       stopUC
                [8] IMPLICIT StopUnitControl-Request
ELSE
,       stopUC
                [8] IMPLICIT NULL
ENDIF
IF ( createUnitControl )
,       createUC
                [9] IMPLICIT CreateUnitControl-Request
ELSE
,       createUC
                [9] IMPLICIT NULL
ENDIF
IF ( addToUnitControl )
,       addToUC
                [10] IMPLICIT AddToUnitControl-Request
ELSE
,       addToUC
                [10] IMPLICIT NULL
ENDIF
IF ( removeFromUnitControl )
,       removeFromUC
                [11] IMPLICIT RemoveFromUnitControl-Request
ELSE
,       removeFromUC
                [11] IMPLICIT NULL
ENDIF
IF ( getUnitControlAttributes )
,       getUCAttributes
                [12] IMPLICIT GetUnitControlAttributes-Request
ELSE
,       getUCAttributes
                [12] IMPLICIT NULL
ENDIF
IF ( loadUnitControlFromFile )
,       loadUCFromFile
                [13] IMPLICIT LoadUnitControlFromFile-Request
ELSE
,       loadUCFromFile
                [13] IMPLICIT NULL
ENDIF
IF ( storeUnitControlToFile )
,       storeUCToFile
                [14] IMPLICIT StoreUnitControlToFile-Request
ELSE
,       storeUCToFile
                [14] IMPLICIT NULL
ENDIF
IF ( deleteUnitControl )
,       deleteUC
                [15] IMPLICIT DeleteUnitControl-Request
ELSE
,       deleteUC
                [15] IMPLICIT NULL
ENDIF
```

```
IF ( defineEventConditionList )
,       defineECL
[16] DefineEventConditionList-Request
ELSE
, defineECL
                [16] IMPLICIT NULL
ENDIF
IF ( deleteEventConditionList )
,       deleteECL
                [17] DeleteEventConditionList-Request
ELSE
,       deleteECL
                [17] IMPLICIT NULL
ENDIF
IF ( addEventConditionListReference )
,       addECLReference
                [18] IMPLICIT AddEventConditionListReference-Request
ELSE
,       addECLReference
                [18] IMPLICIT NULL
ENDIF
IF ( removeEventConditionListReference )
,       removeECLReference
                [19] IMPLICIT RemoveEventConditionListReference-Request
ELSE
,       removeECLReference
                [19] IMPLICIT NULL
ENDIF
IF ( getEventConditionListAttributes )
,       getECLAttributes
                [20] GetEventConditionListAttributes-Request
ELSE
,       getECLAttributes
                [20] IMPLICIT NULL
ENDIF
IF ( reportEventConditionListStatus )
,       reportECLStatus
                [21] IMPLICIT ReportEventConditionListStatus-Request
ELSE
,       reportECLStatus
                [21] IMPLICIT NULL
ENDIF
IF ( alterEventConditionListMonitoring )
,       alterECLMonitoring
                [22] IMPLICIT AlterEventConditionListMonitoring-Request
ELSE
,       alterECLMonitoring
                [22] IMPLICIT NULL
ENDIF
ELSE
,       initiateUCLoad
                [4] IMPLICIT NULL,
        uCLoad
                [5] IMPLICIT NULL,
        uCUpload
```

```
                [6] IMPLICIT NULL,
        startUC
                [7] IMPLICIT NULL,
        stopUC
                [8] IMPLICIT NULL,
        createUC
                [9] IMPLICIT NULL,
        addToUC
                [10] IMPLICIT NULL,
        removeFromUC
                [11] IMPLICIT NULL,
        getUCAttributes
                [12] IMPLICIT NULL,
        loadUCFromFile
                [13] IMPLICIT NULL,
        storeUCToFile
                [14] IMPLICIT NULL,
        deleteUC
                [15] IMPLICIT NULL,
        defineECL
                [16] IMPLICIT NULL,
        deleteECL
                [17] IMPLICIT NULL,
        addECLReference
                [18] IMPLICIT NULL,
        removeECLReference
                [19] IMPLICIT NULL,
        getECLAttributes
                [20] IMPLICIT NULL,
        reportECLStatus
                [21] IMPLICIT NULL,
        alterECLMonitoring
                [22] IMPLICIT NULL
ENDIF
        }
```

7.1.3 Request-Detail (подробности запроса)

```
Request-Detail ::= CHOICE {
                        -- this choice shall be selected if the tag value of the
                        -- ConfirmedServiceRequest does not match any of the tags below
        otherRequests NULL
IF ( createProgramInvocation )
,       createProgramInvocation
                [38] IMPLICIT CS-CreateProgramInvocation-Request
ELSE
,       createProgramInvocation
                [38] IMPLICIT NULL
ENDIF
IF ( start )
,       start
                [40] IMPLICIT CS-Start-Request
ELSE
,       start
                [40] IMPLICIT NULL
ENDIF
```

```
IF ( resume )
,       resume
                [42] IMPLICIT CS-Resume-Request
ELSE
,       resume
                [42] IMPLICIT NULL
ENDIF
IF ( defineEventCondition )
,       defineEventCondition
                [47] IMPLICIT CS-DefineEventCondition-Request
ELSE
,       defineEventCondition
                [47] IMPLICIT NULL
ENDIF
IF ( alterEventConditionMonitoring )
,       alterEventConditionMonitoring
                [51] IMPLICIT CS-AlterEventConditionMonitoring-Request
ELSE
,       alterEventConditionMonitoring
                [51] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
,       defineEventEnrollment
                [57] IMPLICIT CS-DefineEventEnrollment-Request
ELSE
,       defineEventEnrollment
                [57] IMPLICIT NULL
ENDIF
IF ( alterEventEnrollment )
,       alterEventEnrollment
                [59] IMPLICIT CS-AlterEventEnrollment-Request
ELSE
,       alterEventEnrollment
                [59] IMPLICIT NULL
ENDIF
        }
```

7.2 The Unconfirmed-PDU (неподтверждаемый блок данных)

```
Unconfirmed-PDU ::= SEQUENCE {
        service UnconfirmedService,
...
IF (cspi)
, service-ext [79] Unconfirmed-Detail OPTIONAL
ENDIF
                -- shall not be transmitted if value is the value
                -- of a tagged type derived from NULL
        }
```

Неподтверждаемый блок данных **Unconfirmed-PDU** представляет из себя последовательность, содержащую неподтвердаемые услуги **UnconfirmedService** и неподтверждаемые подробности (детали) **Unconfirmed-Detail.**

Блок данных **UnconfirmedService** должен использоваться для идентификации всех существующих неподтверждаемых услуг и их аргументов.

7.2.1 UnconfirmedService (неподтверждаемая услуга)

```
        UnconfirmedService ::= CHOICE {
IF (informationReport)
```

```
        information Report
                [0] IMPLICIT InformationReport
ELSE
        informationReport
                [0] IMPLICIT NULL
ENDIF
IF (unsolicitedStatus)
, unsolicitedStatus
                [1] IMPLICIT UnsolicitedStatus
ELSE
, unsolicitedStatus
                [1] IMPLICIT NULL
ENDIF
IF (eventNotification )
, eventNotification
                [2] IMPLICIT EventNotification
ELSE
, eventNotification
                [2] IMPLICIT NULL
ENDIF
                }
```

Тип блока данных UnconfirmedService должен идентифицировать тип услуги и аргумент для этой услуги. С целью идентификации типа услуги используется соответствующий контекстный тег. Определения для каждой индивидуальной услуги устанавливают форму аргумента услуги с помощью определения типа, который содержит ссылку из блока данных **UnconfirmedService.** Каждая услуга, соответствующая блоку данных **UnconfirmedService** является неподтверждаемой услугой.

7.2.2 Unconfirmed-Detail (неподтверждаемые подробности)

Блок данных подтверждаемого ответа **Confirmed-ResponsePDU** — это последовательность, содержащая три элемента: целое без знака, сущность **ConfirmedServiceResponse** и сущность **Response-Detail**.

```
    Unconfirmed-Detail ::= CHOICE {
                -- this choice shall be selected if the tag value of the
                -- UnconfirmedService does not match any of the tags below
        otherRequests NULL
IF ( cspi )
,       eventNotification
                [2] IMPLICIT CS-EventNotification
ENDIF
        }
```

7.3 Confirmed-ResponsePDU (блок данных подтверждаемого ответа)

```
    Confirmed-ResponsePDU ::= SEQUENCE {
        invokeID                Unsigned32,
        service         ConfirmedServiceResponse,
        ...
IF ( csr cspi ),
        service-ext             [79] Response-Detail OPTIONAL
ENDIF
                                -- shall not be transmitted if value is the value
                                -- of a tagged type derived from NULL
        }
```

Идентификатор задействования **InvokeID** — это 32-битное целое без знака. Он однозначно идентифицирует запрос услуги среди всех ожидающих выполнения подтвержденных запросов услуги от конкретного MMS-пользователя по заданной прикладной ассоциации. В любой момент времени должен иметь место самое большее один ожидающий выполнения запрос услуги от конкретного

MMS-пользователя по некоторой прикладной ассоциации для любого заданного идентификатора задействования **InvokeID**. Значение **InvokeID** указано MMS-пользователем в примитиве запроса услуг (см. ИСО 9506-1, раздел 5). Значение **InvokeID**, указанное в **Confirmed-ResponsePDU** и **Confirmed-ErrorPDU,** предоставляет возможность MMS-провайдеру и MMS-пользователю коррелировать указанные PDU с рассматриваемым запросом услуги.

Сущность **ConfirmedServiceResponse** нужна для идентификации подтверждаемой услуги и ответа по данной подтверждаемой услуги. Данный параметр описан далее в 7.3.1.

7.3.1 ConfirmedServiceResponse (ответ на подтвержденную услугу)

```
    ConfirmedServiceResponse ::= CHOICE {
IF ( status )
        status
                [0] IMPLICIT Status-Response
ELSE
        status
                [0] IMPLICIT RejectPDU
ENDIF
IF ( getNameList )
,       getNameList
                [1] IMPLICIT GetNameList-Response
ELSE
,       getNameList
                [1] IMPLICIT RejectPDU
ENDIF
IF ( identify )
,       identify
                [2] IMPLICIT Identify-Response
ELSE
,       identify
                [2] IMPLICIT RejectPDU
ENDIF
IF ( rename )
,       rename
                [3] IMPLICIT Rename-Response
ELSE
,       rename
                [3] IMPLICIT RejectPDU
ENDIF
IF ( read )
,       read
                [4] IMPLICIT Read-Response
ELSE
,       read
                [4] IMPLICIT RejectPDU
ENDIF
IF ( write )
,       write
                [5] IMPLICIT Write-Response
ELSE
,       write
                [5] IMPLICIT RejectPDU
ENDIF
IF ( vnam vadr )
IF ( getVariableAccessAttributes )
,       getVariableAccessAttributes
```

```
                [6] IMPLICIT GetVariableAccessAttributes-Response
ELSE
,       getVariableAccessAttributes
                [6] IMPLICIT RejectPDU
ENDIF
ELSE
,       getVariableAccessAttributes
                [6] IMPLICIT RejectPDU
ENDIF
IF ( vnam )
IF ( vadr )
IF ( defineNamedVariable )
,       defineNamedVariable
                [7] IMPLICIT DefineNamedVariable-Response
ELSE
,       defineNamedVariable
                [7] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineNamedVariable
                [7] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineNamedVariable
                [7] IMPLICIT RejectPDU
ENDIF
IF ( vsca )
        -- choice [8] is reserved for a service defined in Annex E
IF ( defineScatteredAccess )
,       defineScatteredAccess
                [8] IMPLICIT DefineScatteredAccess-Response
ELSE
,       defineScatteredAccess
                [8] IMPLICIT RejectPDU
ENDIF
        -- choice [9] is reserved for a service defined in Annex E
IF ( getScatteredAccessAttributes )
,       getScatteredAccessAttributes
                [9] IMPLICIT GetScatteredAccessAttributes-Response
ELSE
,       getScatteredAccessAttributes
                [9] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineScatteredAccess
                [8] IMPLICIT RejectPDU,
        getScatteredAccessAttributes
                [9] IMPLICIT RejectPDU
ENDIF
IF ( vnam )
IF ( deleteVariableAccess )
,       deleteVariableAccess
                [10] IMPLICIT DeleteVariableAccess-Response
ELSE
,       deleteVariableAccess
```

```
                [10] IMPLICIT RejectPDU
ENDIF
ELSE
,       deleteVariableAccess
                [10] IMPLICIT RejectPDU
ENDIF
IF ( vlis )
IF ( vnam )
IF ( defineNamedVariableList )
,       defineNamedVariableList
                [11] IMPLICIT DefineNamedVariableList-Response
ELSE
,       defineNamedVariableList
                [11] IMPLICIT RejectPDU
ENDIF
IF ( getNamedVariableListAttributes )
,       getNamedVariableListAttributes
                [12] IMPLICIT GetNamedVariableListAttributes-Response
ELSE
,       getNamedVariableListAttributes
                [12] IMPLICIT RejectPDU
ENDIF
IF ( deleteNamedVariableList )
,       deleteNamedVariableList
                [13] IMPLICIT DeleteNamedVariableList-Response
ELSE
,       deleteNamedVariableList
                [13] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineNamedVariableList
                [11] IMPLICIT RejectPDU,
        getNamedVariableListAttributes
                [12] IMPLICIT RejectPDU,
        deleteNamedVariableList
                [13] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineNamedVariableList
                [11] IMPLICIT RejectPDU,
        getNamedVariableListAttributes
                [12] IMPLICIT RejectPDU,
        deleteNamedVariableList
                [13] IMPLICIT RejectPDU
ENDIF
IF ( vnam )
IF ( defineNamedType )
,       defineNamedType
                [14] IMPLICIT DefineNamedType-Response
ELSE
,       defineNamedType
                [14] IMPLICIT RejectPDU
ENDIF
IF ( getNamedTypeAttributes )
,       getNamedTypeAttributes
```

```
                [15] IMPLICIT GetNamedTypeAttributes-Response
ELSE
,       getNamedTypeAttributes
                [15] IMPLICIT RejectPDU
ENDIF
IF ( deleteNamedType )
,       deleteNamedType
                [16] IMPLICIT DeleteNamedType-Response
ELSE
,       deleteNamedType
                [16] IMPLICIT RejectPDU
ENDIF
ELSE
,       defineNamedType
                [14] IMPLICIT RejectPDU,
        getNamedTypeAttributes
                [15] IMPLICIT RejectPDU,
        deleteNamedType
                [16] IMPLICIT RejectPDU
ENDIF
IF ( input )
,       input
                [17] IMPLICIT Input-Response
ELSE
,       input
                [17] IMPLICIT RejectPDU
ENDIF
IF ( output )
,       output
                [18] IMPLICIT Output-Response
ELSE
,       output
                [18] IMPLICIT RejectPDU
ENDIF
IF ( takeControl )
,       takeControl
                [19] TakeControl-Response
ELSE
,       takeControl
                [19] IMPLICIT RejectPDU
ENDIF
IF ( relinquishControl )
,       relinquishControl
                [20] IMPLICIT RelinquishControl-Response
ELSE
,       relinquishControl
                [20] IMPLICIT RejectPDU
ENDIF
IF ( defineSemaphore )
,       defineSemaphore
                [21] IMPLICIT DefineSemaphore-Response
ELSE
,       defineSemaphore
                [21] IMPLICIT RejectPDU
ENDIF
```

```
IF ( deleteSemaphore )
,       deleteSemaphore
                [22] IMPLICIT DeleteSemaphore-Response
ELSE
,       deleteSemaphore
                [22] IMPLICIT RejectPDU
ENDIF
IF ( reportSemaphoreStatus )
,       reportSemaphoreStatus
                [23] IMPLICIT ReportSemaphoreStatus-Response
ELSE
,       reportSemaphoreStatus
                [23] IMPLICIT RejectPDU
ENDIF
IF ( reportPoolSemaphoreStatus )
,       reportPoolSemaphoreStatus
                [24] IMPLICIT ReportPoolSemaphoreStatus-Response
ELSE
,       reportPoolSemaphoreStatus
                [24] IMPLICIT RejectPDU
ENDIF
IF ( reportSemaphoreEntryStatus )
,       reportSemaphoreEntryStatus
                [25] IMPLICIT ReportSemaphoreEntryStatus-Response
ELSE
,       reportSemaphoreEntryStatus
                [25] IMPLICIT RejectPDU
ENDIF
IF ( initiateDownloadSequence )
,       initiateDownloadSequence
                [26] IMPLICIT InitiateDownloadSequence-Response,
        downloadSegment
                [27] IMPLICIT DownloadSegment-Response,
        terminateDownloadSequence
                [28] IMPLICIT TerminateDownloadSequence-Response
ELSE
,       initiateDownloadSequence
                [26] IMPLICIT RejectPDU,
        downloadSegment
                [27] IMPLICIT RejectPDU,
        terminateDownloadSequence
                [28] IMPLICIT RejectPDU
ENDIF
IF ( initiateUploadSequence )
,       initiateUploadSequence
                [29] IMPLICIT InitiateUploadSequence-Response,
        uploadSegment
                [30] IMPLICIT UploadSegment-Response,
        terminateUploadSequence
                [31] IMPLICIT TerminateUploadSequence-Response
ELSE
,       initiateUploadSequence
                [29] IMPLICIT RejectPDU,
        uploadSegment
                [30] IMPLICIT RejectPDU,
```

```
        terminateUploadSequence
                [31] IMPLICIT RejectPDU
ENDIF
IF ( requestDomainDownload )
,       requestDomainDownload
                [32] IMPLICIT RequestDomainDownload-Response
ELSE
,       requestDomainDownload
                [32] IMPLICIT RejectPDU
ENDIF
IF ( requestDomainUpload )
,       requestDomainUpload
                [33] IMPLICIT RequestDomainUpload-Response
ELSE
,       requestDomainUpload
                [33] IMPLICIT RejectPDU
ENDIF
IF ( loadDomainContent )
,       loadDomainContent
                [34] IMPLICIT LoadDomainContent-Response
ELSE
,       loadDomainContent
                [34] IMPLICIT RejectPDU
ENDIF
IF ( storeDomainContent )
,       storeDomainContent
                [35] IMPLICIT StoreDomainContent-Response
ELSE
,       storeDomainContent
                [35] IMPLICIT RejectPDU
ENDIF
IF ( deleteDomain )
,       deleteDomain
                [36] IMPLICIT DeleteDomain-Response
ELSE
,       deleteDomain
                [36] IMPLICIT RejectPDU
ENDIF
IF ( getDomainAttributes )
,       getDomainAttributes
                [37] IMPLICIT GetDomainAttributes-Response
ELSE
,       getDomainAttributes
                [37] IMPLICIT RejectPDU
ENDIF
IF ( createProgramInvocation )
,       createProgramInvocation
                [38] IMPLICIT CreateProgramInvocation-Response
ELSE
,       createProgramInvocation
                [38] IMPLICIT RejectPDU
ENDIF
IF ( deleteProgramInvocation )
,       deleteProgramInvocation
                [39] IMPLICIT DeleteProgramInvocation-Response
```

```
ELSE
,       deleteProgramInvocation
                [39] IMPLICIT RejectPDU
ENDIF
IF ( start )
,       start
                [40] IMPLICIT Start-Response
ELSE
,       start
                [40] IMPLICIT RejectPDU
ENDIF
IF ( stop )
,       stop
                [41] IMPLICIT Stop-Response
ELSE
,       stop
                [41] IMPLICIT RejectPDU
ENDIF
IF ( resume )
,       resume
                [42] IMPLICIT Resume-Response
ELSE
,       resume
                [42] IMPLICIT RejectPDU
ENDIF
IF ( reset )
,       reset
                [43] IMPLICIT Reset-Response
ELSE
,       reset
                [43] IMPLICIT RejectPDU
ENDIF
IF ( kill )
,       kill
                [44] IMPLICIT Kill-Response
ELSE
,       kill
                [44] IMPLICIT RejectPDU
ENDIF
IF ( getProgramInvocationAttributes )
,       getProgramInvocationAttributes
                [45] IMPLICIT GetProgramInvocationAttributes-Response
ELSE
,       getProgramInvocationAttributes
                [45] IMPLICIT RejectPDU
ENDIF
IF ( obtainFile )
,       obtainFile
                [46] IMPLICIT ObtainFile-Response
ELSE
,       obtainFile
                [46] IMPLICIT RejectPDU
ENDIF
IF ( defineEventCondition )
,       defineEventCondition
```

```
                [47] IMPLICIT DefineEventCondition-Response
ELSE
,       defineEventCondition
                [47] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventCondition )
,       deleteEventCondition
                [48] IMPLICIT DeleteEventCondition-Response
ELSE
,       deleteEventCondition
                [48] IMPLICIT RejectPDU
ENDIF
IF ( getEventConditionAttributes )
,       getEventConditionAttributes
                [49] IMPLICIT GetEventConditionAttributes-Response
ELSE
,       getEventConditionAttributes
                [49] IMPLICIT RejectPDU
ENDIF
IF ( reportEventConditionStatus )
,       reportEventConditionStatus
                [50] IMPLICIT ReportEventConditionStatus-Response
ELSE
,       reportEventConditionStatus
                [50] IMPLICIT RejectPDU
ENDIF
IF ( alterEventConditionMonitoring )
,       alterEventConditionMonitoring
                [51] IMPLICIT AlterEventConditionMonitoring-Response
ELSE
,       alterEventConditionMonitoring
                [51] IMPLICIT RejectPDU
ENDIF
IF ( triggerEvent )
,       triggerEvent
                [52] IMPLICIT TriggerEvent-Response
ELSE
,       triggerEvent
                [52] IMPLICIT RejectPDU
ENDIF
IF ( defineEventAction )
,       defineEventAction
                [53] IMPLICIT DefineEventAction-Response
ELSE
,       defineEventAction
                [53] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventAction )
,       deleteEventAction
                [54] IMPLICIT DeleteEventAction-Response
ELSE
,       deleteEventAction
                [54] IMPLICIT RejectPDU
ENDIF
IF ( getEventActionAttributes )
```

```
,       getEventActionAttributes
                [55] IMPLICIT GetEventActionAttributes-Response
ELSE
,       getEventActionAttributes
                [55] IMPLICIT RejectPDU
ENDIF
IF ( reportEventActionStatus )
,       reportEventActionStatus
                [56] IMPLICIT ReportEventActionStatus-Response
ELSE
,       reportEventActionStatus
                [56] IMPLICIT RejectPDU
ENDIF
IF ( defineEventEnrollment )
,       defineEventEnrollment
                [57] IMPLICIT DefineEventEnrollment-Response
ELSE
,       defineEventEnrollment
                [57] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventEnrollment )
,       deleteEventEnrollment
                [58] IMPLICIT DeleteEventEnrollment-Response
ELSE
,       deleteEventEnrollment
                [58] IMPLICIT RejectPDU
ENDIF
IF ( alterEventEnrollment )
,       alterEventEnrollment
                [59] IMPLICIT AlterEventEnrollment-Response
ELSE
,       alterEventEnrollment
                [59] IMPLICIT RejectPDU
ENDIF
IF ( reportEventEnrollmentStatus )
,       reportEventEnrollmentStatus
                [60] IMPLICIT ReportEventEnrollmentStatus-Response
ELSE
,       reportEventEnrollmentStatus
                [60] IMPLICIT RejectPDU
ENDIF
IF ( getEventEnrollmentAttributes )
,       getEventEnrollmentAttributes
                [61] IMPLICIT GetEventEnrollmentAttributes-Response
ELSE
,       getEventEnrollmentAttributes
                [61] IMPLICIT RejectPDU
ENDIF
IF ( acknowledgeEventNotification )
,       acknowledgeEventNotification
                [62] IMPLICIT AcknowledgeEventNotification-Response
ELSE
,       acknowledgeEventNotification
                [62] IMPLICIT RejectPDU
ENDIF
```

```
IF ( getAlarmSummary )
,       getAlarmSummary
                [63] IMPLICIT GetAlarmSummary-Response
ELSE
,       getAlarmSummary
                [63] IMPLICIT RejectPDU
ENDIF
IF ( getAlarmEnrollmentSummary )
,       getAlarmEnrollmentSummary
                [64] IMPLICIT GetAlarmEnrollmentSummary-Response
ELSE
,       getAlarmEnrollmentSummary
                [64] IMPLICIT RejectPDU
ENDIF
IF ( readJournal )
,       readJournal
                [65] IMPLICIT ReadJournal-Response
ELSE
,       readJournal
                [65] IMPLICIT RejectPDU
ENDIF
IF ( writeJournal )
,       writeJournal
                [66] IMPLICIT WriteJournal-Response
ELSE
,       writeJournal
                [66] IMPLICIT RejectPDU
ENDIF
IF ( initializeJournal )
,       initializeJournal
                [67] IMPLICIT InitializeJournal-Response
ELSE
,       initializeJournal
                [67] IMPLICIT RejectPDU
ENDIF
IF ( reportJournalStatus )
,       reportJournalStatus
                [68] IMPLICIT ReportJournalStatus-Response
ELSE
,       reportJournalStatus
                [68] IMPLICIT RejectPDU
ENDIF
IF ( createJournal )
,       createJournal
                [69] IMPLICIT CreateJournal-Response
ELSE
,       createJournal
                [69] IMPLICIT RejectPDU
ENDIF
IF ( deleteJournal )
,       deleteJournal
                [70] IMPLICIT DeleteJournal-Response
ELSE
,       deleteJournal
                [70] IMPLICIT RejectPDU
```

```
ENDIF
IF ( getCapabilityList )
,           getCapabilityList
                    [71] IMPLICIT GetCapabilityList-Response
ELSE
,           getCapabilityList
                    [71] IMPLICIT RejectPDU
ENDIF
            -- choices [72] through [77] are reserved for use by services
            -- defined in annex D
IF ( fileOpen )
,           fileOpen
                    [72] IMPLICIT FileOpen-Response
ELSE
,           fileOpen
                    [72] IMPLICIT RejectPDU
ENDIF
IF ( fileRead )
,           fileRead
                    [73] IMPLICIT FileRead-Response
ELSE
,           fileRead
                    [73] IMPLICIT RejectPDU
ENDIF
IF ( fileClose )
,           fileClose
                    [74] IMPLICIT FileClose-Response
ELSE
,           fileClose
                    [74] IMPLICIT RejectPDU
ENDIF
IF ( fileRename )
,           fileRename
                    [75] IMPLICIT FileRename-Response
ELSE
,           fileRename
                    [75] IMPLICIT RejectPDU
ENDIF
IF ( fileDelete )
,           fileDelete
                    [76] IMPLICIT FileDelete-Response
ELSE
,           fileDelete
                    [76] IMPLICIT RejectPDU
ENDIF
IF ( fileDirectory )
,           fileDirectory
                    [77] IMPLICIT FileDirectory-Response
ELSE
,           fileDirectory
                    [77] IMPLICIT RejectPDU
ENDIF
,...
IF ( csr cspi )
,           additionalService
```

```
                [78] AdditionalService-Response
                -- choice [79] is reserved
IF ( getDataExchangeAttributes ),
,       getDataExchangeAttributes
                [80] GetDataExchangeAttributes-Response
                        -- Shall not appear in minor version 1
ENDIF
IF ( exchangeData ),
,       exchangeData
                [81] IMPLICIT ExchangeData-Response
                        -- Shall not appear in minor version 1
ENDIF
IF ( defineAccessControlList ),
,       defineAccessControlList
                [82] IMPLICIT DefineAccessControlList-Response
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( getAccessControlListAttributes ),
,       getAccessControlListAttributes
                [83] IMPLICIT GetAccessControlListAttributes-Response
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( reportAccessControlledObjects ),
,       reportAccessControlledObjects
                [84] IMPLICIT ReportAccessControlledObjects-Response
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( deleteAccessControlList ),
,       deleteAccessControlList
                [85] IMPLICIT DeleteAccessControlList-Response
                        -- Shall not appear in minor version 1 or 2
ENDIF
IF ( changeAccessControl ),
,       changeAccessControl
                [86] IMPLICIT ChangeAccessControl-Response
                        -- Shall not appear in minor version 1 or 2
ENDIF
        ,...
        }
```

Тип **ConfirmedServiceResponse** идентифицирует тип услуги и ответ на данную услугу. Теги контекста идентифицируют тип услуги. Определения индивидуальных услуг описывают форму ответа на услугу путем определения типа, на который произведена ссылка в разработке сущности ответа подтверждаемой услуги **ConfirmedServiceResponse**.

7.3.2 AdditionalService-Response (ответ на дополнительную услугу)

```
   AdditionalService-Response ::= CHOICE {
IF ( csr )
IF ( vMDStop )
        vMDStop
                [0] IMPLICIT VMDStop-Response
ELSE
        vMDStop
                [0] IMPLICIT RejectPDU
ENDIF
IF ( vMDReset )
```

```
,       vMDReset
                [1] IMPLICIT VMDReset-Response
ELSE
,       vMDReset
                [1] IMPLICIT RejectPDU
ENDIF
IF ( select )
,       select
                [2] IMPLICIT Select-Response
ELSE
,       select
                [2] IMPLICIT RejectPDU
ENDIF
IF ( alterProgramInvocationAttributes )
,       alterPI
                [3] IMPLICIT AlterProgramInvocationAttributes-Response
ELSE
,       alterPI
                [3] IMPLICIT RejectPDU
ENDIF
ELSE
,       vMDStop
                [0] IMPLICIT RejectPDU,
        vMDReset
                [1] IMPLICIT RejectPDU,
        select
                [2] IMPLICIT RejectPDU,
        alterPI
                [3] IMPLICIT RejectPDU
ENDIF
IF ( cspi )
IF ( initiateUnitControlLoad )
,       initiateUCLoad
                [4] IMPLICIT InitiateUnitControlLoad-Response
ELSE
,       initiateUCLoad
                [4] IMPLICIT RejectPDU
ENDIF
IF ( unitControlLoadSegment )
,       uCLoad
                [5] IMPLICIT UnitControlLoadSegment-Response
ELSE
,       uCLoad
                [5] IMPLICIT RejectPDU
ENDIF
IF ( unitControlUpload )
,       uCUpload
                [6] IMPLICIT UnitControlUpload-Response
ELSE
,       uCUpload
                [6] IMPLICIT RejectPDU
ENDIF
IF ( startUnitControl )
,       startUC
                [7] IMPLICIT StartUnitControl-Response
```

```
ELSE
,       startUC
                [7] IMPLICIT RejectPDU
ENDIF
IF ( stopUnitControl )
,       stopUC
                [8] IMPLICIT StopUnitControl-Response
ELSE
,       stopUC
                [8] IMPLICIT RejectPDU
ENDIF
IF ( createUnitControl )
,       createUC
                [9] IMPLICIT CreateUnitControl-Response
ELSE
,       createUC
                [9] IMPLICIT RejectPDU
ENDIF
IF ( addToUnitControl )
,       addToUC
                [10] IMPLICIT AddToUnitControl-Response
ELSE
,       addToUC
                [10] IMPLICIT RejectPDU
ENDIF
IF ( removeFromUnitControl )
,       removeFromUC
                [11] IMPLICIT RemoveFromUnitControl-Response
ELSE
,       removeFromUC
                [11] IMPLICIT RejectPDU
ENDIF
IF ( getUnitControlAttributes )
,       getUCAttributes
                [12] IMPLICIT GetUnitControlAttributes-Response
ELSE
,       getUCAttributes
                [12] IMPLICIT RejectPDU
ENDIF
IF ( loadUnitControlFromFile )
,       loadUCFromFile
                [13] IMPLICIT LoadUnitControlFromFile-Response
ELSE
,       loadUCFromFile
                [13] IMPLICIT RejectPDU
ENDIF
IF ( storeUnitControlToFile )
,       storeUCToFile
                [14] IMPLICIT StoreUnitControlToFile-Response
ELSE
,       storeUCToFile
                [14] IMPLICIT RejectPDU
ENDIF
IF ( deleteUnitControl )
,       deleteUC
```

```
                    [15] IMPLICIT DeleteUnitControl-Response
ELSE
,           deleteUC
                    [15] IMPLICIT RejectPDU
ENDIF
IF ( defineEventConditionList )
,           defineECL
                    [16] IMPLICIT DefineEventConditionList-Response
ELSE
,           defineECL
                    [16] IMPLICIT RejectPDU
ENDIF
IF ( deleteEventConditionList )
,           deleteECL
                    [17] IMPLICIT DeleteEventConditionList-Response
ELSE
,           deleteECL
                    [17] IMPLICIT RejectPDU
ENDIF
IF ( addEventConditionListReference )
,           addECLReference
                    [18] IMPLICIT AddEventConditionListReference-Response
ELSE
,           addECLReference
                    [18] IMPLICIT RejectPDU
ENDIF
IF ( removeEventConditionListReference )
,           removeECLReference
                    [19] IMPLICIT RemoveEventConditionListReference-Response
ELSE
,           removeECLReference
                    [19] IMPLICIT RejectPDU
ENDIF
IF ( getEventConditionListAttributes )
,           getECLAttributes
                    [20] IMPLICIT GetEventConditionListAttributes-Response
ELSE
,           getECLAttributes
                    [20] IMPLICIT RejectPDU
ENDIF
IF ( reportEventConditionListStatus )
,           reportECLStatus
                    [21] IMPLICIT ReportEventConditionListStatus-Response
ELSE
,           reportECLStatus
                    [21] IMPLICIT RejectPDU
ENDIF
IF ( alterEventConditionListMonitoring )
,           alterECLMonitoring
                    [22] IMPLICIT AlterEventConditionListMonitoring-Response
ELSE
,           alterECLMonitoring
                    [22] IMPLICIT RejectPDU
ENDIF
ELSE
```

```
,       initiateUCLoad
                [4] IMPLICIT RejectPDU,
        uCLoad
                [5] IMPLICIT RejectPDU,
        uCUpload
                [6] IMPLICIT RejectPDU,
        startUC
                [7] IMPLICIT RejectPDU,
        stopUC
                [8] IMPLICIT RejectPDU,
        createUC
                [9] IMPLICIT RejectPDU,
        addToUC
                [10] IMPLICIT RejectPDU,
        removeFromUC
                [11] IMPLICIT RejectPDU,
        getUCAttributes
                [12] IMPLICIT RejectPDU,
        loadUCFromFile
                [13] IMPLICIT RejectPDU,
        storeUCToFile
                [14] IMPLICIT RejectPDU,
        deleteUC
                [15] IMPLICIT RejectPDU,
        defineECL
                [16] IMPLICIT RejectPDU,
        deleteECL
                [17] IMPLICIT RejectPDU,
        addECLReference
                [18] IMPLICIT RejectPDU,
        removeECLReference
                [19] IMPLICIT RejectPDU,
        getECLAttributes
                [20] IMPLICIT RejectPDU,
        reportECLStatus
                21] IMPLICIT RejectPDU,
        alterECLMonitoring
                [22] IMPLICIT RejectPDU
ENDIF
        }
```

7.3.3 Response-Detail (подробности ответа)

```
    Response-Detail ::= CHOICE {
                -- this choice shall be selected if the tag value of the
                -- ConfirmedServiceResponse does not match any of the tags below
        otherRequests           NULL
IF ( status )
,       status
            [0] CS-Status-Response
ENDIF
IF ( getProgramInvocationAttributes )
,       getProgramInvocationAttributes
            [45] IMPLICIT CS-GetProgramInvocationAttributes-Response
ENDIF
IF ( getEventConditionAttributes )
```

```
,       getEventConditionAttributes
            [49] IMPLICIT CS-GetEventConditionAttributes-Response
ENDIF
        }
```

7.4 Confirmed-ErrorPDU (блок данных протокола подтверждаемой ошибки)

```
    Confirmed-ErrorPDU ::= SEQUENCE     {
        invokeID                                [0] IMPLICIT Unsigned32,
IF ( attachToEventCondition attachToSemaphore )
        modifierPosition                [1] IMPLICIT Unsigned32 OPTIONAL,
ENDIF
        serviceError                            [2] IMPLICIT ServiceError
        }
```

Сущность **Confirmed-ErrorPDU —** это последовательность трех элементов: целое без знака, целое без знака по выбору и сущность **ServiceError** (ошибка услуги).

Идентификатор задействования **InvokeID —** это 32-битное целое без знака. Он однозначно идентифицирует запрос услуги среди всех ожидающих выполнения подтвержденных запросов услуги от конкретного MMS-пользователя по заданной прикладной ассоциации. В любой момент времени должен иметь место самое большее один ожидающий выполнения запрос услуги от конкретного MMS-пользователя по некоторой прикладной ассоциации для любого заданного идентификатора задействования **InvokeID**. Значение **InvokeID** указано MMS-пользователем в примитиве запроса услуг (см. ИСО 9506-1, раздел 5). Значение **InvokeID**, указанное в **Confirmed-ResponsePDU** и **Confirmed-ErrorPDU,** предоставляет возможность MMS-провайдеру и MMS-пользователю коррелировать указанные PDU с рассматриваемым запросом услуги.

Сущность **modifierPosition** — это 32-битное целое без знака. Данная сущность однозначно идентифицирует модификатор среди всех модификаторов, указанных в перечне модификаторов **listOfModifiers** блока данных подтверждаемого запроса **Confirmed-RequestPDU,** идентифицированного сущностью **InvokeID**. Данный параметр получается из подпараметра **modifierPosition** параметра ошибки услуги **ServiceEror** из примитива услуги ответа (см. ИСО 9506-1, раздел 24).

Сущность **ServiceError** идентифицирует класс ошибок и код ошибки как для модификатора подтверждаемой услуги, так и для самой подтверждаемой услуги. Параметр **ServiceError** описан в 7.4.1.

7.4.1 ServiceError (ошибка услуги)

```
ServiceError ::= SEQUENCE   {
    errorClass              [0] CHOICE   {
    vmd-state                       [0] IMPLICIT INTEGER   {
            other                               (0),
            vmd-state-conflict                  (1),
            vmd-operational-problem             (2),
            domain-transfer-problem             (3),
            state-machine-id-invalid    (4)
    } (0..4),
    application-reference   [1] IMPLICIT INTEGER   {
            other                               (0),
            application-unreachable     (1),
            connection-lost             (2),
            application-reference-invalid (3),
            context-unsupported         (4)
    } (0..4),
    Definition                      [2] IMPLICIT INTEGER   {
            other                               (0),
            object-undefined                    (1),
            invalid-address             (2),
            type-unsupported                    (3),
            type-inconsistent                   (4),
            object-exists                       (5),
```

```
                object-attribute-inconsistent   (6)
        } (0..6),
        Resource                [3] IMPLICIT INTEGER {
                other                           (0),
                memory-unavailable              (1),
                processor-resource-unavailable  (2),
                mass-storage-unavailable        (3),
                capability-unavailable          (4),
                capability-unknown              (5)
        } (0..5),
        service                 [4] IMPLICIT INTEGER {
                other (0),
                primitives-out-of-sequence      (1),
                object-state-conflict           (2),
                        -- Value 3 reserved for further definition
                continuation-invalid            (4),
                object-constraint-conflict      (5)
        } (0..5),
        service-preempt         [5] IMPLICIT INTEGER {
                other                           (0),
                timeout                         (1),
                deadlock                        (2),
                cancel                          (3)
        } (0..3),
        time-resolution         [6] IMPLICIT INTEGER {
                other                           (0),
                unsupportable-time-resolution   (1)
        } (0..1),
        access                  [7] IMPLICIT INTEGER {
                other                           (0),
                object-access-unsupported       (1),
                object-non-existent             (2),
                object-access-denied            (3),
                object-invalidated              (4)
        } (0..4),
        initiate                [8] IMPLICIT INTEGER {
                other                           (0),
                        -- Values 1 and 2 are reserved for further definition
                max-services-outstanding-calling-insufficient   (3),
                max-services-outstanding-called-insufficient    (4),
                service-CBB-insufficient        (5),
                parameter-CBB-insufficient      (6),
                nesting-level-insufficient      (7)
        } (0..7),
        conclude                [9] IMPLICIT INTEGER {
                other                                   (0),
                further-communication-required          (1)
        } (0..1)
IF ( cancel )
,       cancel                  [10] IMPLICIT INTEGER {
                other                           (0),
                invoke-id-unknown               (1),
                cancel-not-possible             (2)
        } (0..2)
ELSE
```

```
,       cancel                  [10] IMPLICIT NULL
ENDIF
IF ( fileOpen fileClose fileRead fileRename fileDelete fileDirectory obtainFile )
,       file                            [11] IMPLICIT INTEGER {
                other                           (0),
                filename-ambiguous      (1),
                file-busy                       (2),
                filename-syntax-error   (3),
                content-type-invalid    (4),
                position-invalid        (5),
                file-access-denied              (6),
                file-non-existent               (7),
                duplicate-filename              (8),
                insufficient-space-in-filestore (9)
        } (0..9)
ELSE
, file [11] IMPLICIT NULL
ENDIF
,       others                  [12] IMPLICIT INTEGER
} ,
        additionalCode          [1] IMPLICIT INTEGER OPTIONAL,
        additionalDescription   [2] IMPLICIT VisibleString OPTIONAL,
        serviceSpecificInfo             [3] CHOICE {
IF ( obtainFile )
        obtainFile                              [0] IMPLICIT ObtainFile-Error
ELSE
        obtainFile                              [0] IMPLICIT NULL
ENDIF
IF ( start )
,       start                           [1] IMPLICIT Start-Error
ELSE
,       start                           [1] IMPLICIT NULL
ENDIF
IF ( stop )
,       stop                            [2] IMPLICIT Stop-Error
ELSE
,       stop                            [2] IMPLICIT NULL
ENDIF
IF ( resume )
,       resume                          [3] IMPLICIT Resume-Error
ELSE
,       resume                          [3] IMPLICIT NULL
ENDIF
IF ( reset )
,       reset                           [4] IMPLICIT Reset-Error
ELSE
,       reset                           [4] IMPLICIT NULL
ENDIF
IF ( deleteVariableAccess )
,       deleteVariableAccess    [5] IMPLICIT DeleteVariableAccess-Error
ELSE
,       deleteVariableAccess    [5] IMPLICIT NULL
ENDIF
IF ( deleteNamedVariableList )
,       deleteNamedVariableList         [6] IMPLICIT DeleteNamedVariableList-Error
ELSE
```

```
,       deleteNamedVariableList                 [6] IMPLICIT NULL
ENDIF
IF ( deleteNamedType )
,       deleteNamedType                         [7] IMPLICIT DeleteNamedType-Error
ELSE
,       deleteNamedType                         [7] IMPLICIT NULL
ENDIF
IF ( defineEventEnrollment )
,       defineEventEnrollment-Error     [8] DefineEventEnrollment-Error
ELSE
,       defineEventEnrollment-Error     [8] IMPLICIT NULL
ENDIF
        -- [9] Reserved for use by annex D
IF ( fileRename )
,       fileRename                              [9] IMPLICIT FileRename-Error
ELSE
,       fileRename                              [9] IMPLICIT NULL
ENDIF
IF ( csr cspi )
,       additionalService                       [10] AdditionalService-Error
ELSE
,       additionalService                       [10] IMPLICIT NULL
ENDIF
IF ( changeAccessControl )
,       changeAccessControl             [11] IMPLICIT ChangeAccessControl-Error
ELSE
        changeAccessControl             [11] IMPLICIT NULL
ENDIF
        } OPTIONAL
    }
```

Тип **ServiceError** идентифицирует класс ошибки и код ошибки. Он доставляет код ошибки, сообщение об ошибке, а также специальную информацию для услуг, требующих дополнительную информацию, передаваемую при возникновении ошибки. Сущность **ErrorClass**, особые значения **ErrorClass**, а также параметры дополнительного кода **additionalCode** и дополнительного описания **additionalDescription** получаются в соответствии с соглашениями (см. 5.5) в настоящем стандарте и определениями раздела 24 ИСО 9506-1.

Выбор класса ошибки **errorClass** основан на выборе подпараметра класса ошибок для параметра типа ошибки **ErrorType**, указанного примитивом услуги ответа. В свою очередь, выбор значения для класса ошибок основан на выборе подпараметра кода ошибки для параметра типа ошибки **ErrorType**, указанного примитивом услуги ответа. Параметры **additionalCode** и **additionalDescription** получены из подпараметров параметра типа ошибки **ErrorType** с тем же именем.

Особая информация услуги **serviceSpecificInformation** отсутствует, если параметр **modifierPosition** присутствует в блоке данных **Confirmed-ErrorPDU**. Если параметр **modifierPosition** не присутствует в блоке данных **Confirmed-ErrorPDU**, то информация **serviceSpecificInformation** может быть получена из других параметров, описанных как подпараметры параметра **Result(–)** для конкретных услуг (если такие особые подпараметры услуг существуют).

Примечание — Особая информация услуги не указывается, если модификатор требует возврата блока данных **Confirmed-ErrorPDU**. Если возврат **Confirmed-ErrorPDU** – это результат ошибки обработки запроса подтверждаемой услуги, то особая информация услуги возвращается, но только для тех услуг, для которых указанная информация удовлетворяет требованиям установленной процедуры предоставления услуг.

7.4.2 AdditionalService-Error (ошибка дополнительной услуги)

```
    AdditionalService-Error ::= CHOICE    {
IF ( defineEventConditionList )
        defineEcl                               [0] DefineEventConditionList-Error
```

```
ELSE
        defineEcl                       [0] IMPLICIT NULL
ENDIF
IF ( addEventConditionListReference )
,       addECLReference                 [1] AddEventConditionListReference-Error
ELSE
,       addECLReference                 [1] IMPLICIT NULL
ENDIF
IF ( removeEventConditionListReference )
,       removeECLReference      [2] RemoveEventConditionListReference-Error
ELSE
,       removeECLReference      [2] IMPLICIT NULL
ENDIF
IF ( initiateUnitControlLoad )
,       initiateUC                      [3] InitiateUnitControl-Error
ELSE
,       initiateUC                      [3] IMPLICIT NULL
ENDIF
IF ( startUnitControl )
,       startUC                         [4] IMPLICIT StartUnitControl-Error
ELSE
,       startUC                         [4] IMPLICIT NULL
ENDIF
IF ( stopUnitControl )
,       stopUC                          [5] IMPLICIT StopUnitControl-Error
ELSE
,       stopUC                          [5] IMPLICIT NULL
ENDIF
IF ( deleteUnitControl )
,       deleteUC                        [6] DeleteUnitControl-Error
ELSE
,       deleteUC                        [6] IMPLICIT NULL
ENDIF
IF ( loadUnitControlFromFile )
,       loadUCFromFile                  [7] LoadUnitControlFromFile-Error
ELSE
,       loadUCFromFile                  [7] IMPLICIT NULL
ENDIF
        }
```

7.5 Типы обычных MMS

В настоящем подразделе определены ряд типов, на которые произведены ссылки в настоящем стандарте.

7.5.1 TimeOfDay

TimeOfDay ::= OCTET STRING (SIZE(4|6))

Тип **TimeOfDay** — это октетная строка **OCTET STRING**. Значение типа **TimeOfDay** может содержать четыре (4) или шесть (6) октетов. Первая форма указывает время как число миллисекунд, прошедших с полуночи текущей даты (дата в значении не содержится). Вторая форма содержит и время, и дату, выраженную относительно 1 января 1984 г. Первые четыре октета содержат значения, указывающие число миллисекунд, прошедших с полуночи текущей даты, для обеих форм. Значение временной области получается путем нумерации битов указанных октетов, начиная с наименее значительного бита последнего октета (как нулевого бита). Нумерация заканчивается наиболее значительным битом первого октета — битом № 31. Каждому биту назначается численное значение, равное 2**N, где *N* — позиция рассматриваемого бита в настоящей последовательности нумерации. Значение времени получается суммированием численных значений, назначенных каждому биту (для битов со значением,

равным единице). Биты с номерами 28–31 всегда равны нулю.

Ниже представлены октеты типа **TimeOfDay**. Если значение содержит дату (6-октетный контент), то представление (в обозначениях **ASN.1 bstring**) имеет вид:

'0 0 0 0 t t t t t t t t t t t t t t t t t t t t t t t t t t t t d d d d d d d d d d d d d d d d'B

Если значение не содержит дату (4-октетный контент), то последние два октета («d...d») опускают. В вышеуказанном представлении **bstring** «t...t» – это относительное число миллисекунд указанного дня. Число миллисекунд для полуночи равно 0. «d...d» — это относительное число дней, начиная с 1 января 1984 г. (когда оно равно 0). Все значения даны в бинарной форме.

Наиболее значительный бит значения рассматриваемой подобласти указан выше в строке **bstring.** Значения битов уменьшаются для последующих битов этой строки.

Если система использует тип **TimeOfDay**, то она должна описывать степень дробления подобласти «t...t» в утверждении согласованности практической реализации протокола (см. раздел 18).

7.5.2 Идентификаторы и целые типы

В настоящем стандарте рассмотрены типы «**Identifier**», «**Integer8**», «**Integer16**», «**Integer32**», «**Unsigned8**», «**Unsigned16**» и «**Unsigned32**». Указанные типы определены следующим образом.

```
   maxIdentifier INTEGER ::= 32
Identifier ::=
IF (char)
        UTF8String (SIZE(1..maxIdentifier))
ELSE
        VisibleString ( FROM
            («A»|»a»|»B»|»b»|»C»|»c»|»D»|»d»|»E»|»e»|»F»|»f»|
             «G»|»g»|»H»|»h»|»I»|»i»|»J»|»j»|»K»|»k»|»L»|»l»|
             «M»|»m»|»N»|»n»|»O»|»o»|»P»|»p»|»Q»|»q»|»R»|»r»|
             «S»|»s»|»T»|»t»|»U»|»u»|»V»|»v»|»W»|»w»|»X»|»x»|
             «Y»|»y»|»Z»|»z»|»$»|»_»|»0»|»1»|»2»|»3»|»4»|»5»|
             «6»|»7»|»8»|»9») ) (SIZE(1..maxIdentifier))
ENDIF
Integer8 ::= INTEGER(-128..127)                 -- range -128 <= i <= 127
Integer16 ::= INTEGER(-32768..32767)            -- range -32,768 <= i <= 32,767
Integer32 ::= INTEGER(-2147483648..2147483647)          -- range -2**31 <= i <= 2**31 – 1
Unsigned8 ::= INTEGER(0..127)                           -- range 0 <= i <= 127
Unsigned16 ::= INTEGER(0..32767)                -- range 0 <= i <= 32767
Unsigned32 ::= INTEGER(0..2147483647)           -- range 0 <= i <= 2**31 – 1
```

Среда MMS определяет различные типы имен (имя переменной, имя типа и т. д.) в терминах разработки идентификатора. Длина идентификатора ограничена 32 символами. Они выбираются из набора символов, определенных либо типом видимой строки **VisibleString** (если символы **char** СВВ не поддерживаются), либо типом **UTF8String** (если символы **char** СВВ поддерживаются). Выбор идентификатора зависит от конкретной ситуации.

Типы **Integer8**, **Integer16**, **Integer32**, **Unsigned8**, **Unsigned16** и **Unsigned32** В настоящем стандарте представлены целые в ограниченном диапазоне. Минимальное и максимальное представительные значения описаны в комментариях, следующих за объявлением типа.

7.5.3 ObjectName (имя объекта)

```
ObjectName ::= CHOICE {
    vmd-specific            [0] IMPLICIT Identifier,
    domain-specific [1] IMPLICIT SEQUENCE {
            domainID                    Identifier,
            itemID              Identifier
            },
    aa-specific             [2] IMPLICIT Identifier
    }
```

Параметр **ObjectName** получается в соответствии с правилами 5.5 на основе определения параметра услуги **ObjectName**, представленного в разделе 7 ИСО 9506-1.

7.5.4 ObjectClass (класс объекта)

```
    ObjectClass ::= CHOICE {
        basicObjectClass                                    [0] IMPLICIT INTEGER {
IF ( vnam )
                namedVariable                   (0)
ENDIF
                                        -- value 1 is reserved for definition in Annex E
IF ( vsca )
,               scatteredAccess                             (1)
ENDIF
IF ( vlis )
,               namedVariableList                           (2)
ENDIF
IF ( vnam )
,               namedType                                   (3)
ENDIF
,               semaphore                                   (4),
                eventCondition                  (5),
                eventAction                                 (6),
                eventEnrollment                             (7),
                journal                         (8),
                domain                          (9),
                programInvocation                           (10),
                operatorStation                 (11),
                dataExchange                    (12),
        -- Shall not appear in minor version 1
                accessControlList                           (13)
        -- Shall not appear in minor version 1 or 2
                } (0..13),
            ...
IF ( cspi )
,       csObjectClass           [1] IMPLICIT INTEGER {
                eventConditionList                          (0),
                unitControl                                 (1) } (0..1)
ENDIF
    }
```

Параметр **ObjectClass** получается в соответствии с правилами 5.5. Определение параметра услуги **ObjectClass** представлено в разделе 7 ИСО 9506-1.

7.5.5 ApplicationReference (ссылка приложения)

Форма типа **ApplicationReference** зависит от используемой системы связи. Для определений, совместимых с OSI-связью, представлены в приложении А.

7.5.6 MMSString (строка MMS)

Строка **MMSString** используется для хранения пользовательских строк в соответствующих наборах символов. Ее тип определен следующим образом:

```
    MMSString ::=
IF ( char )
                UTF8String
ELSE
                VisibleString
ENDIF
    MMS255String ::=
IF ( char )
                UTF8String (SIZE(1..255))
```

ELSE

VisibleString (SIZE(1..255))

ENDIF

Если аргумент **char** не описан, то строка **MMSString** дает известное множество 94 символов, используемое в английском языке. Если аргумент **char** описан, то **MMSString** делает возможным полную спецификацию в соответствии с требованиями **UNICODE**.

7.5.7 FileName (имя файла)

filename ::= SEQUENCE OF GraphicString

Тип **FileName** включает последовательность графических строк. Определение семантики элементов указанной последовательности графических строк имени файла — это локальная тема. Любые ограничения, наложенные системой, удовлетворяющей требованиям ИСО 9506-1 и настоящего стандарта, на длину и корректные обозначения сущности **FileName,** описаны утверждением конфигурации и инициализации системы (см. раздел 25). Как минимум, каждая практическая реализация, использующая тип **FileName,** определенный в настоящем пункте, должна поддерживать имена файлов, содержащие один элемент, включающий от одной до восьми заглавных букв или цифр. Имя всегда начинается с буквы.

Примечание — ИСО 9506-1 и настоящий стандарт не дают интерпретацию компонент сущности **FileName.** Указанные компоненты доставляют прозрачный механизм поименования заказчику и ответчику. Соотношение между компонентами, определенное в виртуальном файлохранилище, и любое деление сущности на компоненты в реальной среде функционирования системы — это локальный выбор практической реализации. Рассматриваемая практическая реализация может отображать локальную структуру компонентов на компоненты сущности **FileName.** Также она может отображать существующий синтаксис **FileName** на сущность **FileName** с однокомпонентным именем. Практическая реализация может отражать компоненты **FileName** среды MMS путем выбора пути доступа к реальному файлу. Однако данный выбор не всегда очевиден при обеспечении взаимосвязи в среде MMS.

8 Среда и протокол общего управления

8.1 Введение

Настоящий раздел описывает блоки данных PDU услуг, формирующих среду, и услуги общего управления. Настоящий раздел содержит описание протокола реализации следующих услуг:

инициирование;
завершение;
прерывание;
отмена;
выбраковка.

8.2 Инициирование

Абстрактный синтаксис запроса (ответа, ошибки) инициирования услуги описан типами **Initiate-RequestPDU**, **Initiate-ResponsePDU** и **Initiate-ErrorPDU** соответственно, представленными ниже. В 5.5 описан порядок получения всех параметров, не приведенных явно в настоящем разделе. Любые дополнительные корректные помеченные значения ASN.1, полученные как элементы последовательности **Initiate-RequestPDU**, **Initiate-ResponsePDU** и **Initiate-ErrorPDU**, игнорируются для обеспечения требуемого уровня совместимости.

```
Initiate-RequestPDU ::= SEQUENCE {
  localDetailCalling                    [0] IMPLICIT Integer32 OPTIONAL,
  proposedMaxServOutstandingCalling     [1] IMPLICIT Integer16,
  proposedMaxServOutstandingCalled      [2] IMPLICIT Integer16,
  proposedDataStructureNestingLevel     [3] IMPLICIT Integer8 OPTIONAL,
  initRequestDetail                     [4] IMPLICIT SEQUENCE {
      proposedVersionNumber                 [0] IMPLICIT Integer16,
      proposedParameterCBB                  [1] IMPLICIT ParameterSupportOptions,
      servicesSupportedCalling              [2] IMPLICIT ServiceSupportOptions ,
      ...
```

```
IF (csr cspi)
,       additionalSupportedCalling              [3] IMPLICIT AdditionalSupportOptions
ENDIF
IF (cspi)
,       additionalCbbSupportedCalling           [4] IMPLICIT AdditionalCBBOptions,
        privilegeClassIdentityCalling           [5] IMPLICIT VisibleString
ENDIF
        }
    }
  Initiate-ResponsePDU ::= SEQUENCE {
    localDetailCalled                           [0] IMPLICIT Integer32 OPTIONAL,
    negotiatedMaxServOutstandingCalling         [1] IMPLICIT Integer16,
    negotiatedMaxServOutstandingCalled          [2] IMPLICIT Integer16,
    negotiatedDataStructureNestingLevel         [3] IMPLICIT Integer8 OPTIONAL,
    initResponseDetail                          [4] IMPLICIT SEQUENCE {
        negotiatedVersionNumber                 [0] IMPLICIT Integer16,
        negotiatedParameterCBB                  [1] IMPLICIT ParameterSupportOptions,
        servicesSupportedCalled                 [2] IMPLICIT ServiceSupportOptions,
        ...
IF (csr cspi)
,       additionalSupportedCalled               [3] IMPLICIT AdditionalSupportOptions
ENDIF
IF (cspi )
,       additionalCbbSupportedCalled            [4] IMPLICIT AdditionalCBBOptions,
        privilegeClassIdentityCalled            [5] IMPLICIT VisibleString
ENDIF
        }
    }
  Initiate-ErrorPDU ::= ServiceError
```

8.2.1 Initiate-RequestPDU (блок данных запроса инициирования)

Абстрактный синтаксис запроса инициирования услуги — это **Initiate-RequestPDU**.

8.2.2 Initiate-ResponsePDU (блок данных ответ инициирования)

Абстрактный синтаксис ответа услуги инициирования — это **Initiate-ResponsePDU**.

8.2.3 Initiate-ErrorPDU (блок данных ошибки инициирования)

Абстрактный синтаксис ошибки услуги инициирования — это **Initiate-ErrorPDU**.

8.3 Завершение

Астрактный синтаксис запроса (ответа, ошибки) услуги завершения описан типами **Conclude-RequestPDU**, **Conclude-ResponsePDU** и **Conclude-ErrorPDU** соответственно, приведенными ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных в настоящем подразделе.

```
Conclude-RequestPDU ::= NULL
Conclude-ResponsePDU ::= NULL
Conclude-ErrorPDU ::= ServiceError
```

8.3.1 Conclude-RequestPDU (блок данных запроса завершения)

Абстрактный синтаксис запроса услуги завершения — это **Conclude-RequestPDU**.

8.3.2 Conclude-ResponsePDU (блок данных ответа завершения)

Абстрактный синтаксис услуги ответа завершения — это **Conclude-ResponsePDU**.

8.3.3 Conclude-ErrorPDU (блок данных ошибки завершения)

Абстрактный синтаксис ошибки услуги завершения — это **Conclude-ErrorPDU**.

8.4 Прерывание

Услуга прерывания прямо отображается на услугу **M-U-Abort** (см. раздел 24).

8.5 Отмена

Абстрактный синтаксис запроса (ответа, ошибки) услуги отмены описан типами **Cancel-RequestPDU, Cancel-ResponsePDU** и **Cancel-ErrorPDU** соответственно, приведенными ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных в настоящем подразделе.

```
Cancel-RequestPDU ::= Unsigned32 – originalInvokeID
Cancel-ResponsePDU ::= Unsigned32 – originalInvokeID
Cancel-ErrorPDU ::= SEQUENCE {
    originalInvokeID            [0] IMPLICIT Unsigned32,
    serviceError                [1] IMPLICIT ServiceError
}
```

8.5.1 Cancel-RequestPDU (блок данных запроса отмены)

Абстрактный синтаксис запроса услуги отмены — это **Cancel-RequestPDU**.

8.5.2 Cancel-ResponsePDU (блок данных ответа отмены)

Абстрактный синтаксис ответа услуги отмены — это **Cancel-ResponsePDU**.

8.5.3 Cancel-ErrorPDU (блок данных ошибки отмены)

Абстрактный синтаксис ошибки услуги отмены — это **Cancel-ErrorPDU**.

8.6 Выбраковка

Описание абстрактного синтаксиса услуги выбраковки определено сущностью **RejectPDU**. В 5.5 представлено описание порядка получения всех параметров, не описанных в настоящем подразделе.

```
RejectPDU ::= SEQUENCE    {
    originalInvokeID                [0] IMPLICIT Unsigned32 OPTIONAL,
    rejectReason                    CHOICE    {
        confirmed-requestPDU        [1] IMPLICIT INTEGER    {
            other                                   (0),
            unrecognized-service                    (1),
            unrecognized-modifier                   (2),
            invalid-invokeID                        (3),
            invalid-argument                        (4),
            invalid-modifier                        (5),
            max-serv-outstanding-exceeded           (6),
                                    -- Value 7 reserved for further definition
            max-recursion-exceeded                  (8),
            value-out-of-range                      (9)
            } (0..9),
        confirmed-responsePDU [2] IMPLICIT INTEGER {
            other                                   (0),
            unrecognized-service                    (1),
            invalid-invokeID                        (2),
            invalid-result                          (3),
                                    -- Value 4 reserved for further definition
            max-recursion-exceeded                  (5),
            value-out-of-range                      (6)
            } (0..6),
        confirmed-errorPDU [3] IMPLICIT INTEGER {
            other                                   (0),
            unrecognized-service                    (1),
            invalid-invokeID                        (2),
            invalid-serviceError                    (3),
```

```
                value-out-of-range                          (4)
                } (0..4),
        unconfirmedPDU [4] IMPLICIT INTEGER {
                other                                       (0),
                unrecognized-service                (1),
                invalid-argument                            (2),
                max-recursion-exceeded                      (3),
                value-out-of-range                          (4)
                } (0..4),
        pdu-error [5] IMPLICIT INTEGER {
                unknown-pdu-type                            (0),
                invalid-pdu                                 (1),
                illegal-acse-mapping                (2)
                },
IF ( cancel )
        cancel-requestPDU [6] IMPLICIT INTEGER {
                other                                       (0),
                invalid-invokeID                    (1)
                } (0..1),
        cancel-responsePDU [7] IMPLICIT INTEGER {
                other                                       (0),
                invalid-invokeID                    (1)
                } (0..1),
        cancel-errorPDU [8] IMPLICIT INTEGER {
                other                                       (0),
                invalid-invokeID                    (1),
                invalid-serviceError                        (2),
                value-out-of-range                          (3)
                } (0..3),
ELSE
        cancel-requestPDU [6] IMPLICIT NULL,
        cancel-responsePDU [7] IMPLICIT NULL,
        cancel-errorPDU [8] IMPLICIT NULL,
ENDIF
        conclude-requestPDU [9] IMPLICIT INTEGER {
                other                                       (0),
                invalid-argument                            (1)
                } (0..1),
        conclude-responsePDU [10] IMPLICIT INTEGER {
                other                                       (0),
                invalid-result                              (1)
                } (0..1),
        conclude-errorPDU [11] IMPLICIT INTEGER {
                other                                       (0),
                invalid-serviceError                        (1),
                value-out-of-range                          (2)
                } (0..2)
        }
}
```

Абстрактный синтаксис услуги выбраковки — это **RejectPDU**. Область причины выбраковки **RejectReason** получается из типа **RejectPDU** и параметров **RejectCode** спецификации услуги. Сделанный выбор должен соответствовать типу **RejectPDU**, указанному в параметре услуги в соответствии с имеющимися комментариями. Выбранное значение должно соответствовать значению кода отказа, указанному в параметре услуги в соответствии с имеющимися комментариями.

9 Протокол ответа на услугу, удовлетворяющую заданным требованиям

9.1 Введение

В настоящем подразделе приведено описание протокола, необходимого для реализации услуг, определенных в разделе 9 ИСО 9506-1. Перечень услуг:

DefineAccessControlList
GetAccessControlListAttributes
ReportAccessControlledObjects
DeleteAccessControlList
ChangeAccessControl

9.2 Условие доступа

Абстрактный синтаксис параметра условия доступа — это тип **AccessCondition**. Тип **AccessCondition** определен в 9.1.2 ИСО 9506-1.

9.3 DefineAccessControlList (перечень средств управления порядком доступа)

Описание абстрактного синтаксиса выбора перечня **DefineAccessControlList** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** задано типами запроса **DefineAccessControlList-Request** и ответа **DefineAccessControlList-Response** соответственно, представленными ниже. В 5.5 представлено описание порядка получения всех параметров, не указанных в настоящем подразделе.

```
DefineAccessControlList-Request ::= SEQUENCE {
    accessControlListName               [0] IMPLICIT Identifier,
    accessControlListElements                   [1] IMPLICIT SEQUENCE {
        readAccessCondition                         [0] AccessCondition OPTIONAL,
        storeAccessCondition                        [1] AccessCondition OPTIONAL,
        writeAccessCondition                        [2] AccessCondition OPTIONAL,
        loadAccessCondition                         [3] AccessCondition OPTIONAL,
        executeAccessCondition              [4] AccessCondition OPTIONAL,
        deleteAccessCondition               [5] AccessCondition OPTIONAL,
        editAccessCondition                         [6] AccessCondition OPTIONAL
        }
    }
DefineAccessControlList-Response ::= NULL
```

9.3.1 DefineAccessControlList-Request (запрос перечня средств управления порядком доступа)

Абстрактный синтаксис выбора **DefineAccessControlList-Request** типа запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineAccessControlList-Request**.

9.3.1.1 AccessControlListElements (элементы перечня средств управления доступом)

Область **AccessControlListElements** — это параметр элементов перечня средств управления доступом примитива запроса **DefineAccessControlList.request.** Он проявляется как параметр переченя элементов управления доступом отображения **DefineAccessControlList.indication**. Данная область содержит не более одной реализации типа **AccessCondition** для каждого из семи возможных значений параметра класса услуг. Условие доступа **AccessCondition,** ассоциированное с классом услуг (равным сущности **Read**), указано в области **ReadAccessCondition** элементов перечня **AccessControlListElements** (аналогичный порядок установлен для других значений параметра класса услуг).

9.3.2 DefineAccessControlList-Response (ответ перечня средств управления порядком доступа)

Абстрактный синтаксис выбора ответа **DefineAccessControlList-Response** для типа ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineAccessControlList-Response**.

9.4 GetAccessControlListAttributes (атрибуты перечня средств управления получением доступа)

Описание абстрактного синтаксиса выбора атрибутов **GetAccessControlListAttributes** запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами запроса **GetAccessControlListAttributes-Request** и ответа **GetAccessControlListAttributes-Response** соответственно, указанными ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных в настоящем подразделе.

```
GetAccessControlListAttributes-Request ::= CHOICE {
        accessControlListName                          [0] IMPLICIT Identifier,
        vMD                                            [1] IMPLICIT NULL,
        namedObject                                [2] IMPLICIT SEQUENCE {
              objectClass                                    [0] ObjectClass,
              objectName                                     [1] ObjectName
              }
}
GetAccessControlListAttributes-Response ::= SEQUENCE {
      name                        [0] Identifier,
      accessControlListElements   [1] IMPLICIT SEQUENCE {
            readAccessCondition              [0] AccessCondition OPTIONAL,
            storeAccessCondition             [1] AccessCondition OPTIONAL,
            writeAccessCondition             [2] AccessCondition OPTIONAL,
            loadAccessCondition              [3] AccessCondition OPTIONAL,
            executeAccessCondition       [4] AccessCondition OPTIONAL,
            deleteAccessCondition        [5] AccessCondition OPTIONAL,
            editAccessCondition              [6] AccessCondition OPTIONAL
            },
      vMDuse                             [2] IMPLICIT BOOLEAN,
      references                         [3] IMPLICIT SEQUENCE OF SEQUENCE {
            objectClass                      [0] ObjectClass,
            objectCount                      [1] IMPLICIT INTEGER
            }
IF ( aco )
,     accessControlList                  [4] IMPLICIT Identifier OPTIONAL
                              -- shall be included if and only if
                              -- aco has been negotiated
ENDIF
      }
```

9.4.1 GetAccessControlListAttributes-Request (запрос атрибутов перечня средств управления получением доступа)

Абстрактный синтаксис запроса арибутов **GetAccessControlListAttributes-Request** типа **ConfirmedServiceRequest** — это **GetAccessControlListAttributes-Request**.

9.4.2 GetAccessControlListAttributes-Response (ответ получения атрибутов перечня средств управления доступом)

Абстрактный синтаксис ответа атрибутов **GetAccessControlListAttributes-Response** типа **ConfirmedServiceResponse —** это **GetAccessControlListAttributes-Response**.

9.4.2.1 accessControlListElements (элементы перечня средств управления доступом)

Область **AccessControlListElements** — это параметр элементов перечня средств управления доступом для примитива ответа **DefineAccessControlList.response.** Он указан как параметр элементов перечня средств управления доступом примитива подтверждения **DefineAccessControlList.confirm**. Данная область содержит не более одной реализации типа **AccessCondition** для каждого из семи возможных значений параметра класса услуг. Условие доступа **AccessCondition,** ассоциированное с классом услуг (равным сущности **Read**), указано в области **readAccessCondition** для элементов перечня **accessControlListElements** (аналогично другим значениям параметра класса услуг).

9.4.2.2 Подсчет управляемых объектов

Абстрактный синтаксис подсчета параметра управляемых объектов — это область ссылок **References** для типа **GetAccessControlListAttributes-Response**. Для каждого класса объектов, содержащего один или несколько объектов, ссылающихся на указанный объект перечня средств управления доступом, следует рассматривать последовательность **objectClass** и **objectCount.** Если в классе нет объектов, ссылающихся на указанный объект перечня средств управления доступом (результат подсчета — нуль), то данная последовательность не рассматривается. Если атрибут **aco** не оговорен, то вышеуказанная область отсутствует.

9.4.2.3 Перечень средств управления доступом

Параметр **AccessControlList** используется, если атрибут **aco CBB** оговорен.

9.5 ReportAccessControlledObjects (отчет об объектах с управлением доступа)

Описание абстрактного синтаксиса выбора объектов **ReportAccessControlObjects** запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** задано типами запроса **ReportAccessControlledObjects-Request** и ответа **ReportAccessControlledObjects-Response**, соответственно. Указанные типы описаны ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных в настоящем подразделе.

```
ReportAccessControlledObjects-Request ::= SEQUENCE {
    accessControlList                   [0] IMPLICIT Identifier,
    objectClass                         [1] ObjectClass,
    continueAfter                       [2] ObjectName OPTIONAL
    }
ReportAccessControlledObjects-Response ::= SEQUENCE {
    listOfNames                         [0] IMPLICIT SEQUENCE OF ObjectName,
    moreFollows                         [1] IMPLICIT BOOLEAN DEFAULT FALSE
    }
```

9.5.1 ReportAccessControlledObjects-Request (запрос отчета об объектах с управляемым доступом)

Абстрактный синтаксис выбора запроса **ReportAccessControlledObjects-Request** для типа **ConfirmedServiceRequest** — это **ReportAccessControlledObjects-Request**.

9.5.2 ReportAccessControlledObjects-Response (ответ отчета об объектах с управляемым доступом)

Абстрактный синтаксис выбора ответа **ReportAccessControlObjects-Response** для типа **ConfirmedServiceResponse** — это ответ **ReportAccessControlledObjects-Response**.

9.6 DeleteAccessControlList (удаление перечня средств управления доступом)

Описание абстрактного синтаксиса выбора перечня **DeleteAccessControlList** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** задано типом запроса **DeleteAccessControlList-Request** и типом ответа **DeleteAccessControlList-Response** соответственно. Указанные типы описаны ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteAccessControlList-Request ::= Identifier
            -- Name of Access Control List Object
DeleteAccessControlList-Response ::= NULL
```

9.6.1 DeleteAccessControlList-Request (запрос удаления перечня средств управления доступом)

Абстрактный синтаксис выбора запроса **DeleteAccessControlList-Request** для типа **ConfirmedServiceRequest** — это запрос **DeleteAccessControlList-Request**.

9.6.2 DeleteAccessControlList-Response (ответ удаления перечня средств управления доступом)

Абстрактный синтаксис выбора ответа **DeleteAccessControlList-Response** для типа **ConfirmedServiceResponse** — это ответ **DeleteAccessControlList-Response**.

9.7 ChangeAccessControl (изменение управлением доступа)

Описание абстрактного синтаксиса выбора **ChangeAccessControl** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** задано типом запроса **ChangeAccessControl-Request** и типом ответа **ChangeAccessControl-Response**, соответственно. Указанные типы описаны ниже. В 5.5 представлено описание порядка получения всех параметров, не описанных явно в настоящем подразделе.

```
ChangeAccessControl-Request ::= SEQUENCE {
    scopeOfChange           CHOICE {
```

```
        vMDOnly                     [0] IMPLICIT NULL,
        listOfObjects               [1] IMPLICIT SEQUENCE {
            objectClass                     [0] ObjectClass,
            objectScope                     [1] CHOICE {
                specific                            [0] IMPLICIT SEQUENCE OF ObjectName,
                                    -- Names of the objects (of class objectClass)
                                    -- whose access is to be changed
                aa-specific                 [1] IMPLICIT NULL,
                domain                              [2] IMPLICIT Identifier,
                                    -- Name of the Domain whose elements
                                    -- are to be changed
                vmd                         [3] IMPLICIT NULL
                }
            }
        },
    accessControlListName           [2] IMPLICIT Identifier
                -- name of the AccessControlList Object that contains
                -- the conditions for access control
    }
ChangeAccessControl-Response ::= SEQUENCE {
    numberMatched           [0] IMPLICIT Unsigned32,
    numberChanged           [1] IMPLICIT Unsigned32
    }
ChangeAccessControl-Error ::= Unsigned32
```

9.7.1 ChangeAccessControl-Request (запрос изменения управлением доступа)

Абстрактный синтаксис выбора запроса **ChangeAccessControl-Request** для типа **ConfirmedServiceRequest** — это запрос **ChangeAccessControl-Request**.

9.7.2 ChangeAccessControl-Response (ответ изменения управлением доступа)

Абстрактный синтаксис выбора ответа **ChangeAccessControl-Response** для типа **ConfirmedServiceResponse** — это ответ **ChangeAccessControl-Response**.

9.7.3 ChangeAccessControl-Error (ошибка изменения управлением доступа)

Абстрактный синтаксис выбора органа управления **ChangeAccessControl** для типа ошибки услуги **ServiceError** — это ошибка изменения управлением доступа **ChangeAccessControl-Error**.

10 Протокол поддержки VMD

10.1 Введение

Настоящий раздел содержит описания блоков данных PDU услуг поддержки VMD. Данный раздел содержит описание протокола, необходимого для реализации следующих услуг:

Status	Rename
UnsolicitedStatus	GetCapabilityList
GetNameList	VMDStop
Identify	VMDReset

10.2 Параметр ответа статуса

Абстрактный синтаксис параметра ответа статуса — это тип **StatusResponse**.

```
StatusResponse ::= SEQUENCE {
    vmdLogicalStatus            [0] IMPLICIT INTEGER {
        state-changes-allowed               (0),
        no-state-changes-allowed            (1),
        limited-services-permitted          (2),
        support-services-allowed            (3)
```

```
        } (0..3),
    vmdPhysicalStatus              [1] IMPLICIT INTEGER {
        operational                                (0),
        partially-operational              (1),
        inoperable                                 (2),
        needs-commissioning                (3)
        } (0..3),
    localDetail                    [2] IMPLICIT BIT STRING (SIZE(0..128)) OPTIONAL
    }
```

10.2.1 CS-Status-Response

```
  CS-Status-Response ::= CHOICE {
IF ( csr )
      fullResponse          SEQUENCE {
        operationState                 [0] IMPLICIT OperationState,
        extendedStatus                 [1] IMPLICIT ExtendedStatus,
        extendedStatusMask             [2] IMPLICIT ExtendedStatus DEFAULT '1111'B,
        selectedProgramInvocation      CHOICE {
            programInvocation                      [3] IMPLICIT Identifier,
            noneSelected                   [4] IMPLICIT NULL   }   }
ENDIF
IF ( csr cspi )
,
ENDIF
IF ( cspi )
  noExtraResponse NULL
ENDIF
  }
```

10.2.2 OperationState (рабочее состояние)

```
OperationState ::= INTEGER {
    idle                                   (0),
    loaded                                 (1),
    ready                                  (2),
    executing                  (3),
    motion-paused                          (4),
    manualInterventionRequired             (5) } (0..5)
```

10.2.3 ExtendedStatus (расширенный статус)

```
ExtendedStatus ::= BIT STRING {
    safetyInterlocksViolated       (0),
    anyPhysicalResourcePowerOn     (1),
    allPhysicalResourcesCalibrated (2),
    localControl                           (3) } (SIZE(4))
```

10.3 Статус

Описание абстрактного синтаксиса выбора статуса для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **Status-Request** и **Status-Response** соответственно. Указанные типы описаны ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Status-Request ::= BOOLEAN -- Extended Derivation
Status-Response ::= StatusResponse
```

10.3.1 Status-Request (запрос статуса)

Абстрактный синтаксис выбора запроса статуса **ConfirmedServiceRequest** — это **Status-Request**.

10.3.2 Status-Response (ответ статуса)

Абстрактный синтаксис выбор ответа статуса **ConfirmedServiceResponse** — это **Status-Response**.

10.4 UnsolicitedStatus (незапрашиваемый статус)

Абстрактный синтаксис выбора **unsolicitedStatus** неподтверждаемой услуги **UnconfirmedService** описан типом **UnsolicitedStatus**, представленным ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
UnsolicitedStatus := StatusResponse
```

10.4.1 UnsolicitedStatus (незапрашиваемый статус)

Абстрактный синтаксис выбора **unsolicitedStatus** неподтверждаемой услуги **UnconfirmedService** — это **UnsolicitedStatus**.

10.5 GetNameList (получение перечня имен)

Описание абстрактного синтаксиса выбора **GetNameList** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** задается типами **GetNameList-Request** и **GetNameList-Response** соответственно, приведенными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetNameList-Request ::= SEQUENCE {
    objectClass                 [0] ObjectClass,
    objectScope                 [1] CHOICE {
        vmdSpecific                 [0] IMPLICIT NULL,
        domainSpecific                  [1] IMPLICIT Identifier,
        aaSpecific                      [2] IMPLICIT NULL },
        continueAfter       [2] IMPLICIT Identifier OPTIONAL }
GetNameList-Response ::= SEQUENCE {
    listOfIdentifier            [0] IMPLICIT SEQUENCE OF Identifier,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

10.5.1 GetNameList-Request (запрос получения перечня имен)

Абстрактный синтаксис выбора **GetNameList** для запроса **ConfirmedServiceRequest** — это **GetNameList-Request**.

10.5.1.1 Область применения объекта

Особый **vmdSpecific** выбор в рамках запроса **GetNameList-Request** производится в том случае, если значение параметра области применения объекта примитива запроса услуги удовлетворяет требованиям виртуального приспособления **VMD**.

Особый **domainSpecific** выбор в рамках запроса **GetNameList-Request** производится в том случае, если значение параметра области применения объекта примитива запроса услуги удовлетворяет требованиям **Domain.** Значение типа **domainSpecific** получается из значения параметра имени домена примитива запроса услуги.

Особый **aaSpecific** выбор в рамках запроса **GetNameList-Request** производится в том случае, если значение параметра области применения объекта примитива запроса услуги имеет тип **AA-Specific**.

10.5.2 GetNameList-Response (ответ получения перечня имен)

Абстрактный синтаксис выбора перечня **GetNameList** ответа **ConfirmedServiceResponse** — это **GetNameList-Response**.

10.6 Identify (идентификация)

Описание абстрактного синтаксиса выбора **identify** запроса **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Identify-Request** и **Identify-Response**, соответственно, приведенными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Identify-Request ::= NULL
Identify-Response ::= SEQUENCE {
```

```
vendorName                  [0] IMPLICIT MMSString,
    modelName          [1] IMPLICIT MMSString,
revision                    [2] IMPLICIT MMSString,
listOfAbstractSyntaxes      [3] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER OPTIONAL
}
```

10.6.1 Identify-Request (запрос идентификации)

Абстрактный синтаксис выбора **identify** запроса **ConfirmedServiceRequest** — это **Identify-Request**.

10.6.2 Identify-Response (ответ идентификации)

Абстрактный синтаксис выбора **identify** ответа **ConfirmedServiceResponse** — это **Identify-Response**.

10.7 Rename (переименование)

Описание абстрактного синтаксиса выбора **rename** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Rename-Request** и **Rename-Response**, соответственно, представленными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Rename-Request ::= SEQUENCE {
    objectClass                 [0] ObjectClass,
    currentName                 [1] ObjectName,
    newIdentifier               [2] IMPLICIT Identifier }
Rename-Response ::= NULL
```

10.7.1 Rename-Request (запрос переименования)

Абстрактный синтаксис выбора **rename** запроса **ConfirmedServiceRequest** — это **Rename-Request**.

10.7.2 Rename-Response (ответ переименования)

Абстрактный синтаксис выбора **rename** ответа **ConfirmedServiceResponse** — это **Rename-Response**.

10.8 GetCapabilityList (получение перечня возможностей)

Описание абстрактного синтаксиса выбора **GetCapabilityList** запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **GetCapabilityList-Request** и **GetCapabilityList-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetCapabilityList-Request ::= SEQUENCE {
    continueAfter           MMSString OPTIONAL
    }
GetCapabilityList-Response ::= SEQUENCE {
    listOfCapabilities              [0] IMPLICIT SEQUENCE OF MMSString,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE
    }
```

10.8.1 GetCapabilityList-Request (запрос получения перечня возможностей)

Абстрактный синтаксис выбора **GetCapabilityList** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetCapabilityList-Request.**

10.8.2 GetCapabilityList-Response (ответ получения перечня возможностей)

Абстрактный синтаксис выбора **GetCapabilityList** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetCapabilityList-Response**.

10.9 VMDStop (остановка виртуального производственного устройства)

Описание абстрактного синтаксиса выбора **VMDStop** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** задано типами **VMDStop-Request** и **VMDStop-Response**, соответственно. Указанные типы описаны ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

VMDStop-Request ::= NULL
VMDStop-Response ::= NULL

10.9.1 VMDStop-Request

Абстрактный синтаксис выбора **VMDStop** запроса **ConfirmedServiceRequest** — это **VMDStop-Request**.

10.9.2 VMDStop-Response

Абстрактный синтаксис выбора **VMDStop** ответа **ConfirmedServiceResponse** — это **VMDStop-Response**.

10.10 VMDReset (перезагрузка виртуального производственного устройства)

Описание абстрактного синтаксиса выбора перезагрузки **VMDReset** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **VMDReset-Request** и **VMDReset-Response**, соответственно. Указанные типы описаны ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

VMDReset-Request ::= BOOLEAN -- Extended Derivation
VMDReset-Response ::= StatusResponse

10.10.1 VMDReset-Request (запрос перезагрузки виртуального производственного устройства)

Абстрактный синтаксис выбора **VMDReset** для запроса **ConfirmedServiceRequest** — это **VMDReset-Request**.

10.10.2 VMDReset-Response (ответ перезагрузки виртуального производственного устройства)

Абстрактный синтаксис выбора **VMDReset** ответа **ConfirmedServiceResponse** — это **VMDReset-Response.**

11 Протокол управления доменом

11.1 Введение

Настоящий раздел содержит описание сервисных **service-Specific** элементов протокола, определенных в части управления доменом определения MMS-услуги. Этими элементами являются:

InitiateDownloadSequence
DownloadSegment
TerminateDownloadSequence
InitiateUploadSequence
UploadSegment
TerminateUploadSequence
RequestDomainDownload
RequestDomainUpload
LoadDomainContent
StoreDomainContent
DeleteDomain
GetDomainAttributes

11.2 InitiateDownloadSequence (последовательность инициирования загрузки)

Описание абстрактного синтаксиса выбора **InitiateDownloadSequence** запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **InitiateDownloadSequence-Request** и **InitiateDownloadSequence-Response**, соответственно. Указанные типы описаны ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
InitiateDownloadSequence-Request ::= SEQUENCE {
        domainName              [0] IMPLICIT Identifier,
        listOfCapabilities              [1] IMPLICIT SEQUENCE OF MMSString,
        sharable                        [2] IMPLICIT BOOLEAN }
InitiateDownloadSequence-Response ::= NULL
```

11.2.1 InitiateDownloadSequence-Request (запрос последовательности инициирования загрузки)

Абстрактный синтаксис выбора **InitiateDownloadSequence** запроса **ConfirmedServiceRequest** — это **InitiateDownloadSequence-Request**.

11.2.2 InitiateDownloadSequence-Response (ответ последовательности инициирования загрузки)

Абстрактный синтаксис выбора **InitiateDownloadSequence** ответа **ConfirmedServiceResponse** — это **InitiateDownloadSequence-Response**.

11.3 DownloadSegment (сегмент загрузки)

Описание абстрактного синтаксиса выбора **DownloadSegment** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **DownloadSegment-Request** и **DownloadSegment-Response,** соответственно. Указанные типы описаны ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DownloadSegment-Request ::= Identifier -- Domain Name
DownloadSegment-Response ::= SEQUENCE {
    loadData                    LoadData,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
LoadData ::= CHOICE {
    non-coded                   [0] IMPLICIT OCTET STRING,
    coded                       EXTERNAL,
    embedded                    EMBEDDED PDV }
```

11.3.1 DownloadSegment-Request (запрос сегмента загрузки)

Абстрактный синтаксис выбора **DownloadSegment** запроса **ConfirmedServiceRequest** — это **DownloadSegment-Request.**

11.3.2 DownloadSegment-Response (ответ сегмента загрузки)

Абстрактный синтаксис выбора **DownloadSegment** ответа **ConfirmedServiceResponse** — это **DownloadSegment-Response.**

11.3.2.1 Загрузка данных

Абстрактный синтаксис параметра загрузки данных ответа услуги **DownloadSegment** — это выбор между октетной строкой **OCTET STRING**, указывающей, что значение данного параметра далее не описано, и его интерпретация — это локальная тема, или что значение имеет тип **EXTERNAL** или **EMBEDDED PDV**, указывающий, что абстрактный синтаксис, на который произведена ссылка описаниями **EXTERNAL** или **EMBEDDED PDV,** содержит правила кодирования для интерпретации значения настоящего параметра.

11.4 TerminateDownloadSequence (завершение последовательности загрузки)

Описание абстрактного синтаксиса выбора **terminateDownloadSequence** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **TerminateDownloadSequence-Request** и **TerminateDownloadSequence-Response,** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
TerminateDownloadSequence-Request ::= SEQUENCE {
    domainName          [0] IMPLICIT Identifier,
    discard                 [1] IMPLICIT ServiceError OPTIONAL }
TerminateDownloadSequence-Response ::= NUL
```

11.4.1 TerminateDownloadSequence-Request (запрос завершения последовательности загрузки)

Абстрактный синтаксис выбора **terminateDownloadSequence** запроса **ConfirmedServiceRequest** — это **TerminateDownloadSequence-Request**.

11.4.1.1 Discard (отбрасывание)

Абстрактный синтаксис параметра **Discard** обеспечен услугой **ServiceError**. Если параметр **Discard** присутствует в запросе услуги **TerminateDownloadSequence,** то тип **ServiceError** должен быть рассмотрен для того, чтобы указать причину отбрасывания. Если параметр **Discard** не присутствует в запросе услуги **TerminateDownloadSequence**, то область **ServiceError** не включается.

11.4.2 TerminateDownloadSequence-Response (ответ завершения последовательности загрузки)

Абстрактный синтаксис выбора **terminateDownloadSequence** ответа **ConfirmedServiceResponse** — это **TerminateDownloadSequence-Response**.

11.5 InitiateUploadSequence (инициирование последовательности подкачки)

Описание абстрактного синтаксиса выбора **InitiateUploadSequence** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **InitiateUploadSequence-Request** и **InitiateUploadSequence-Response,** соответственно. Указанные типы приведены ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
InitiateUploadSequence-Request ::= Identifier -- Domain Name
InitiateUploadSequence-Response ::= SEQUENCE {
    ulsmID                          [0] IMPLICIT Integer32,
    listOfCapabilities              [1] IMPLICIT SEQUENCE OF MMSString }
```

11.5.1 InitiateUploadSequence-Request (запрос инициирования последовательности подгрузки)

Абстрактный синтаксис выбора **InitiateUploadSequence** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **InitiateUploadSequence-Request**.

11.5.2 InitiateUploadSequence-Response (ответ инициирования последовательности подгрузки)

Абстрактный синтаксис выбора **InitiateUploadSequence** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **InitiateUploadSequence-Response**.

11.6 UploadSegment (сегмент подгрузки)

Описание абстрактного синтаксиса выбора **UploadSegment** для запроса **ConfirmedServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **UploadSegment-Request** и **UploadSegment-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
UploadSegment-Request ::= Integer32 -- ULSM ID
UploadSegment-Response ::= SEQUENCE {
    loadData            LoadData,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

11.6.1 UploadSegment-Request (запрос сегмента подгрузки)

Абстрактный синтаксис выбора **UploadSegment** для запроса подтверждаемой услуги **ConfirmedServiceRequest — это UploadSegment-Request**.

11.6.2 UploadSegment-Response (ответ сегмента подгрузки)

Абстрактный синтаксис выбора **UploadSegment** для ответа подтверждаемой услуги **ConfirmedServiceResponse — это UploadSegment-Response.**

11.6.2.1 Загрузка данных

Абстрактный синтаксис параметра загрузки данных ответа услуги **DownloadSegment** — это выбор между октетной строкой **OCTET STRING** (указывающей, что значение данного параметра далее не описано, и его интерпретация имеет локальный характер) или описаниями типа **EXTERNAL** или **EMBEDDED PDV** (указывающими, что абстрактный синтаксис, на который произведена ссылка описаниями **EXTERNAL** или **EMBEDDED PDV,** содержит правила кодирования, необходимые для интерпретации значения данного параметра).

11.7 TerminateUploadSequence (завершение последовательности подгрузки)

Описание абстрактного синтаксиса выбора **TerminateUploadSequence** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **TerminateUploadSequence-Request** и **TerminateUploadSequence-Response,** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
TerminateUploadSequence-Request ::= Integer32 -- ULSM ID
TerminateUploadSequence-Response ::= NULL
```

11.7.1 TerminateUploadSequence-Request (запрос завершения последовательности подгрузки)

Абстрактный синтаксис выбора **TerminateUploadSequence** для запроса **ConfirmedServiceRequest** — это **TerminateUploadSequence-Request**.

11.7.2 TerminateUploadSequence-Response (ответ завершения последовательности подгрузки)

Абстрактный синтаксис выбора **TerminateUploadSequence** для ответа **ConfirmedServiceResponse** — это **TerminateUploadSequence-Response**.

11.8 RequestDomainDownload (загрузка области запроса)

Описание абстрактного синтаксиса выбора **RequestDomainDownload** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **RequestDomainDownload-Request** и **RequestDomainDownload-Response,** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
RequestDomainDownload-Request ::= SEQUENCE {
    domainName          [0] IMPLICIT Identifier,
    listOfCapabilities      [1] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
    sharable                [2] IMPLICIT BOOLEAN,
    fileName                [4] IMPLICIT FileName }
RequestDomainDownload-Response ::= NULL
```

11.8.1 RequestDomainDownload-Request (запрос загрузки области запроса)

Абстрактный синтаксис выбора **RequestDomainDownload** для запроса **ConfirmedServiceRequest** — это **RequestDomainDownload-Request**.

Если параметр перечня возможности присутствует в запросе услуги и данный параметр содержит описание пустого перечня, то передается значение **SEQUENCE OF** с нулевыми элементами. Если указанный параметр не присутствует в запросе услуги, то данная область не передается.

11.8.2 RequestDomainDownload-Response (ответ загрузки области запроса)

Абстрактный синтаксис выбора **RequestDomainDownload** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **RequestDomainDownload-Response**.

11.9 RequestDomainUpload (подгрузка домена запроса)

Описание абстрактного синтаксиса выбора **RequestDomainUpload** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **ConfirmedServiceRequest** и **RequestDomainUpload-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем разделе.

```
RequestDomainUpload-Request ::= SEQUENCE {
    domainName          [0] IMPLICIT Identifier,
    fileName                [1] IMPLICIT FileName }
RequestDomainUpload-Response ::= NULL
```

11.9.1 RequestDomainUpload-Request (запрос подгрузки домена запроса)

Абстрактный синтаксис выбора **RequestDomainUpload** для запроса **ConfirmedServiceRequest** — это **RequestDomainUpload-Request**.

11.9.2 RequestDomainUpload-Response (ответ подгрузки домена запроса)

Абстрактный синтаксис выбора **RequestDomainUpload** ответа подтверждаемой услуги **ConfirmedServiceResponse — это RequestDomainUpload-Response.**

11.10 LoadDomainContent (контент домена загрузки)

Описание абстрактного синтаксиса выбора **LoadDomainContent** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **LoadDomainContent-Request** и **LoadDomainContent-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
LoadDomainContent-Request ::= SEQUENCE {
        domainName          [0] IMPLICIT Identifier,
        listOfCapabilities  [1] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
        sharable            [2] IMPLICIT BOOLEAN,
        fileName            [4] IMPLICIT FileName
IF ( tpy )
,       thirdParty          [5] IMPLICIT ApplicationReference OPTIONAL
ENDIF
        }
LoadDomainContent-Response ::= NULL
```

11.10.1 LoadDomainContent-Request (запрос контента домена загрузки)

Абстрактный синтаксис выбора **LoadDomainContent** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **LoadDomainContent-Request.**

Если параметр перечня возможностей присутствует в запросе услуги и данный параметр содержит описание пустого перечня, то передается значение **SEQUENCE OF** с нулевыми элементами. Если данный параметр не присутствует в запросе услуги, то данная область не передается.

11.10.1.1 ApplicationReference (ссылка приложения)

Абстрактный синтаксис параметра **thirdParty** (третья сторона кроме отправителя и получателя сообщений) услуги **LoadDomainContent** — это **ApplicationReference**. Данный параметр не указывается, если структурный элемент согласованности параметра **tpy** не поддерживается. Если структурный элемент параметра согласованности **tpy** поддерживается, то используется параметр **thirdParty** по выбору.

11.10.2 LoadDomainContent-Response (ответ контента домена загрузки)

Абстрактный синтаксис выбора **LoadDomainContent** ответа **ConfirmedServiceResponse** — это **LoadDomainContent-Response**.

11.11 StoreDomainContent (контент области хранения)

Описание абстрактного синтаксиса выбора **StoreDomainContent** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **StoreDomainContent-Request** и **StoreDomainContent-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
StoreDomainContent-Request ::= SEQUENCE {
        domainName              [0] IMPLICIT Identifier,
        fileName                    [1] IMPLICIT FileName
IF ( tpy )
,       thirdParty                  [2] IMPLICIT ApplicationReference OPTIONAL
ENDIF
        }
StoreDomainContent-Response ::= NULL
```

11.11.1 StoreDomainContent-Request (запрос контента области хранения)

Абстрактный синтаксис выбора **StoreDomainContent** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **StoreDomainContent-Request**.

11.11.1.1 ApplicationReference (ссылка приложения)

Абстрактный синтаксис параметра **thirdParty** услуги **LoadDomainContent** — это **ApplicationReference**. Данный параметр не указывается, если структурный элемент параметра согласованности **tpy** не поддерживается. Если структурный элемент параметра согласованности tpy поддерживается, то используется параметр **thirdParty** по выбору.

11.11.2 StoreDomainContent-Response (ответ контента области хранения)

Абстрактный синтаксис выбора **StoreDomainContent** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **StoreDomainContent-Response**.

11.12 DeleteDomain (стереть область)

Описание абстрактного синтаксиса выбора **DeleteDomain** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **DeleteDomain-Request** и **DeleteDomain-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteDomain-Request ::= Identifier -- Domain Name
DeleteDomain-Response ::= NULL
```

11.12.1 DeleteDomain-Request (запрос удаления области)

Абстрактный синтаксис выбора **DeleteDomain** для запроса **ConfirmedServiceRequest** — это **DeleteDomain-Request**.

11.12.2 DeleteDomain-Response (ответ удаления области)

Абстрактный синтаксис выбора **DeleteDomain** для ответа **ConfirmedServiceResponse** — это **DeleteDomain-Response**.

11.13 GetDomainAttributes (получение атрибутов области)

Описание абстрактного синтаксиса выбора **GetDomainAttributes** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **GetDomainAttributes-Request** и **GetDomainAttributes-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetDomainAttributes-Request ::= Identifier -- Domain Name
GetDomainAttributes-Response ::= SEQUENCE {
        listOfCapabilities              [0] IMPLICIT SEQUENCE OF MMSString,
        state                   [1] IMPLICIT DomainState,
        mmsDeletable                    [2] IMPLICIT BOOLEAN,
        sharable                        [3] IMPLICIT BOOLEAN,
        listOfProgramInvocations        [4] IMPLICIT SEQUENCE OF Identifier,
                                            -- Program Invocation Names
        uploadInProgress                [5] IMPLICIT Integer8
IF ( aco )
,       accessControlList               [6] IMPLICIT Identifier OPTIONAL
                        -- Shall not appear in minor version one or two
ENDIF
        }
```

11.13.1 GetDomainAttributes-Request (запрос получения атрибутов области)

Абстрактный синтаксис выбора **GetDomainAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetDomainAttributes-Request**.

11.13.2 GetDomainAttributes-Response (ответ получения атрибутов области)

Абстрактный синтаксис выбора **GetDomainAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetDomainAttributes-Response**.

11.13.2.1 state (состояние)

Тип **DomainState** определен в разделе 11 ИСО 9506-1.

11.13.2.2 AccessControlList (перечень средств управления доступом)

Область **AccessControlList** указывается только в том случае, если оговорен параметр **aco CBB**.

12 Протокол управления активизацией программы

12.1 Введение

В настоящем разделе приведено описание протокола реализации услуг, определенных в разделе «Управление активизацией программы определения MMS-услуги», в том числе:

CreateProgramInvocation
DeleteProgramInvocation
Start
Stop
Resume
Reset
Kill
GetProgramInvocationAttributes
Select
AlterProgramInvocationAttributes
ReconfigureProgramInvocation

12.2 CreateProgramInvocation (активизация разработки программы)

Описание абстрактного синтаксиса выбора **CreateProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **CreateProgramInvocation-Request** и **CreateProgramInvocation-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
   CreateProgramInvocation-Request ::= SEQUENCE {
         programInvocationName      [0] IMPLICIT Identifier,
         listOfDomainNames          [1] IMPLICIT SEQUENCE OF Identifier,
         reusable                        [2] IMPLICIT BOOLEAN DEFAULT TRUE
IF ( eventNotification )
IF ( getProgramInvocationAttributes )
,        monitorType                [3] IMPLICIT BOOLEAN OPTIONAL
                        -- TRUE indicates PERMANENT monitoring,
                        -- FALSE indicates CURRENT monitoring
ENDIF
ENDIF
        }
   CreateProgramInvocation-Response ::= NULL
   CS-CreateProgramInvocation-Request ::= INTEGER {
         normal (0),
         controlling (1),
         controlled (2)
         } (0..2)
```

12.2.1 CreateProgramInvocation-Request (запрос активизации разработки программы)

Абстрактный синтаксис выбора **CreateProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **CreateProgramInvocation-Request**.

12.2.1.1 Monitor

Абстрактный синтаксис логического параметра **Monitor** услуги **CreateProgramInvocation** зависит от наличия или отсутствия области **MonitorType**. Если область **MonitorType** присутствует, то она должна указывать, что значение параметра **Monitor** равно **true**. Значение области **MonitorType** должно указывать значение параметра **MonitorType** запроса услуги, а также является полученное значение регистрации события **permanent** (постоянным) или **current** (текущим). Если область **MonitorType** отсутствует, то значение параметра **Monitor** равно **false**, и мониторинг не требуется.

12.2.2 CreateProgramInvocation-Response (ответ активизации разработки программы)

Абстрактный синтаксис выбора **CreateProgramInvocation** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **CreateProgramInvocation-Response**.

12.2.3 CS-CreateProgramInvocation-Request (запрос активизации разработки программы)

Абстрактный синтаксис выбора **CreateProgramInvocation** для подробностей запроса **Request-Detail** — это **CS-CreateProgramInvocation-Request.**

12.3 DeleteProgramInvocation (задействование удаления программы)

Описание абстрактного синтаксиса выбора **DeleteProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **DeleteProgramInvocation-Request** и **DeleteProgramInvocation-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteProgramInvocation-Request ::= Identifier -- Program Invocation Name
DeleteProgramInvocation-Response ::= NULL
```

12.3.1 DeleteProgramInvocation-Request (запрос задействования удаления программы)

Абстрактный синтаксис выбора **DeleteProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteProgramInvocation-Request**.

12.3.2 DeleteProgramInvocation-Response (ответ задействования удаления программы)

Абстрактный синтаксис выбора **DeleteProgramInvocation** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DeleteProgramInvocation-Response**.

12.4 Start

Описание абстрактного синтаксиса выбора **Start** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Start-Request** и **Start-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Start-Request ::= SEQUENCE {
    programInvocationName       [0] IMPLICIT Identifier,
    executionArgument                   CHOICE {
        simpleString                    [1] IMPLICIT MMSString,
        encodedString                           EXTERNAL,
        embeddedString                          EMBEDDED PDV    } OPTIONAL    }
Start-Response ::= NULL
Start-Error ::= ProgramInvocationState
CS-Start-Request ::= [0] CHOICE {
    normal              NULL,
    controlling         SEQUENCE {
            startLocation               [0] IMPLICIT VisibleString OPTIONAL,
            startCount          [1] StartCount DEFAULT cycleCount: 1
            }    }
StartCount ::= CHOICE {
    noLimit                             [0] IMPLICIT NULL,
    cycleCount                          [1] IMPLICIT INTEGER,
    stepCount                           [2] IMPLICIT INTEGER }
```

12.4.1 Start-Request (запрос начала)

Абстрактный синтаксис выбора **Start** для запроса **ConfirmedServiceRequest** — это **Start-Request**.

12.4.1.1 Аргумент выполнения

Параметр аргумента выполнения запроса услуги **Start** – это выбор между строкой **MMSString** (указывающей, что далее значение данного параметра не описано, и его интерпретация имеет локальный характер) или описаниями **EXTERNAL** или **EMBEDDED PDV** (указывающими, что абстрактный синтаксис, на который произведена ссылка описаниями **EXTERNAL** или **EMBEDDED PDV**, содержит правила кодирования для интерпретации значения данного параметра).

12.4.2 Start-Response (ответ услуги начала)

Абстрактный синтаксис выбора **Start** для ответа **ConfirmedServiceResponse** — это **Start-Response**.

12.4.3 Start-Error (ошибка услуги начала)

Абстрактный синтаксис выбора **Start** для выбора **serviceSpecificInformation** типа **ConfirmedServiceError** — это параметр **Start-Error**, который является подпараметром состояния вызова программы для параметра **Result(–)** примитива ответа **Start.response**. Данный параметр появляется как подпараметр состояния вызова программы для параметра **Result(–)** примитива подтверждения **Start.confirm** (при его наличии).

12.4.3.1 ProgramInvocationState (состояние задействования программы)

Абстрактный синтаксис поля **ProgramInvocationState** определен в разделе 12 ИСО 9506-1.

12.4.4 CS-Start-Request (запрос начала CS-типа)

Абстрактный синтаксис выбора **Start** для сущности **Request-Detail** — это **CS-Start-Request**.

12.5 Stop

Описание абстрактного синтаксиса выбора **Stop** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Stop-Request** и **Stop-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Stop-Request ::= SEQUENCE {
        programInvocationName                [0] IMPLICIT Identifier }
Stop-Response ::= NULL
Stop-Error ::= ProgramInvocationState
```

12.5.1 Stop-Request (запрос останова)

Абстрактный синтаксис выбора **Stop** для запроса **ConfirmedServiceRequest** — это **Stop-Request.**

12.5.2 Stop-Response (ответ останова)

Абстрактный синтаксис выбора **Stop** для ответа **ConfirmedServiceResponse** — это **Stop-Response**.

12.5.3 Stop-Error (ошибка останова)

Абстрактный синтаксис выбора **Stop** для выбора **ServiceSpecificInformation** типа **ConfirmedServiceError** — это **Stop-Error**, который является подпараметром состояния вызова программы для параметра **Result(–)** примитива ответа **Stop.response.** Он имеет вид как подпараметр состояния вызова программы для параметра **Result(–)** примитива подтверждения **Stop.confirm** (при его наличии).

12.6 Resume (возобновление выполнения)

Описание абстрактного синтаксиса выбора **Resume** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Resume-Request** и **Resume-Response**, соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Resume-Request ::= SEQUENCE    {
        programInvocationName           [0] IMPLICIT Identifier,
        executionArgument               CHOICE {
                simpleString                    [1] IMPLICIT MMSString,
                encodedString           EXTERNAL,
                enmbeddedString                 EMBEDDED PDV } OPTIONAL
        }
Resume-Response ::= NULL
Resume-Error ::= ProgramInvocationState
CS-Resume-Request ::= [0] CHOICE {
        normal                          NULL,
        controlling                     SEQUENCE {
                modeType                        CHOICE {
                continueMode            [0] IMPLICIT NULL,
```

```
changeMode                    [1] StartCount
}     }     }
```

12.6.1 Resume-Request (запрос возобновления выполнения)

Абстрактный синтаксис выбора **Resume** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **Resume-Request**.

12.6.1.1 Аргумент выполнения

Параметр аргумента выполнения запроса услуги **Resume** — это выбор между строкой **MMSString** (указывающей, что значение данного параметра далее не описано, и его интерпретация имеет локальный характер) и описаниями **EXTERNAL** и **EMBEDDED PDV** (указывающими, что абстрактный синтаксис, на который произведена ссылка описаниями **EXTERNAL** и **EMBEDDED PDV**, содержит правила кодирования для интерпретации значений данного параметра).

12.6.2 Resume-Response (ответ возобновления выполнения)

Абстрактный синтаксис выбора **Resume** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **Resume-Response**.

12.6.3 Resume-Error (ошибка возобновления услуги)

Абстрактный синтаксис выбора **Resume** для выбора **ServiceSpecificInformation** типа **ConfirmedServiceError** — это **Resume-Error.** Этот параметр является подпараметром состояния вызова программы для параметра **Result(–)** примитива ответа **Resume.response.** Он имеет вид подпараметра состояния вызова программы для параметра **Result(–)** примитива подтверждения **Resume. confirm** (при его наличии).

12.6.4 CS-Resume-Request (запрос возобновления выполнения типа CS)

Абстрактный синтаксис выбора **Resume** для деталей запроса **Request-Detail** — это **CS-Resume-Request**.

12.7 Reset (перезагрузка)

Описание абстрактного синтаксиса выбора **Reset** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **Reset-Request** и **Reset-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Reset-Request ::= SEQUENCE {
    programInvocationName          [0] IMPLICIT Identifier        }
Reset-Response ::= NULL
Reset-Error ::= ProgramInvocationState
```

12.7.1 Reset-Request (запрос перезагрузки)

Абстрактный синтаксис выбора **Reset** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **Reset-Request**.

12.7.2 Reset-Response (ответ перезагрузки)

Абстрактный синтаксис выбора **Reset** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **Reset-Response**.

12.7.3 Reset-Error (ошибка перезагрузки)

Абстрактный синтаксис выбора **Reset** для выбора **ServiceSpecificInformation** типа **ConfirmedServiceError** — это **Reset-Error**. Этот параметр является подпараметром состояния вызова программы для параметра **Result(–)** примитива ответа **Reset.response**. Он выглядит как подпараметр состояния вызова программы для параметра **Result(–)** примитива подтверждения **Reset.confirm** (при его наличии).

12.8 Kill (аннулирование)

Описание абстрактного синтаксиса выбора **Kill** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено

типами **Kill-Request** и **Kill-Response**, соответственно, указаннымии ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Kill-Request ::= SEQUENCE {
        programInvocationName          [0] IMPLICIT Identifier      }
Kill-Response ::= NULL
```

12.8.1 Kill-Request (запрос аннулирования)

Абстрактный синтаксис выбора **Kill** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **Kill-Request**.

12.8.2 Kill-Response (ответ аннулирования)

Абстрактный синтаксис выбора **Kill** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **Kill-Response**.

12.9 GetProgramInvocationAttributes (получить атрибуты вызова программы)

Описание абстрактного синтаксиса выбора **GetProgramInvocationAttributes** для запроса **Confirmed-ServiceRequest** и ответа **ConfirmedServiceResponse** определено типами **GetProgramInvocationAttributes-Request** и **GetProgramInvocationAttributes-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetProgramInvocationAttributes-Request ::= Identifier -- Program Invocation Name
GetProgramInvocationAttributes-Response ::= SEQUENCE {
    state                       [0] IMPLICIT ProgramInvocationState,
    listOfDomainNames           [1] IMPLICIT SEQUENCE OF Identifier,
    mmsDeletable                [2] IMPLICIT BOOLEAN,
    reusable                    [3] IMPLICIT BOOLEAN,
    monitor                     [4] IMPLICIT BOOLEAN,
    executionArgument           CHOICE {
        simpleString            [5] IMPLICIT MMSString,
        encodedString                   EXTERNAL,
        enmbeddedString                 EMBEDDED PDV }
IF ( aco )
,       accessControlList       [6] IMPLICIT Identifier OPTIONAL
            -- Shall not appear in minor version one or two
ENDIF
      }
CS-GetProgramInvocationAttributes-Response ::= SEQUENCE {
    errorCode                   [0] IMPLICIT INTEGER,
    control                     [1] CHOICE {
        controlling             [0] IMPLICIT SEQUENCE {
            controlledPI        [0] IMPLICIT SEQUENCE OF Identifier,
            programLocation     [1] IMPLICIT VisibleString OPTIONAL,
            runningMode         [2] CHOICE {
                freeRunning     [0] IMPLICIT NULL,
                cycleLimited    [1] IMPLICIT INTEGER,
                stepLimited     [2] IMPLICIT INTEGER }
            },
            controlled          [1] CHOICE {
                controllingPI   [0] IMPLICIT Identifier,
                none            [1] IMPLICIT NULL
            },
            normal              [2] IMPLICIT NULL } }
```

12.9.1 GetProgramInvocationAttributes-Request (запрос получения атрибутов вызова программы)

Абстрактный синтаксис выбора **GetProgramInvocationAttributes** для запроса **ConfirmedServiceRequest** — это **GetProgramInvocationAttributes-Request**.

12.9.2 GetProgramInvocationAttributes-Response (ответ получения атрибутов вызова программы)

Абстрактный синтаксис выбора **GetProgramInvocationAttributes** для ответа **ConfirmedServiceResponse** — это **GetProgramInvocationAttributes-Response**.

12.9.2.1 Аргумент выполнения

Параметр «аргумент выполнения» для ответа услуги **GetProgramInvocationAttributes** — это выбор между строкой **MMSString** (указывающей, что значение данного параметра далее не описано, и его интерпретация имеет локальный характер) и описаниями **EXTERNAL** или **EMBEDDED PDV** (указывающих, что абстрактный синтаксис, на который произведена ссылка описаниями **EXTERNAL** и **EMBEDDED PDV,** содержит правила кодирования для интерпретации значений данного параметра).

12.9.2.2 Перечень средств управления доступом

Параметр **AccessControlList** появляется, если и только если оговорено значение **aco CBB**.

12.9.3 CS-GetProgramInvocationAttributes-Response (ответ получения атрибутов вызова программы типа CS)

Абстрактный синтаксис выбора **GetProgramInvocationAttributes** для услуги **Response-Detail** — это **CS-GetProgramInvocationAttributes-Response.**

12.10 Select (выбор)

Описание абстрактного синтаксиса выбора **Select** для запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **Select-Request** и **Select-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Select-Request ::= SEQUENCE {
        controlling [0] IMPLICIT Identifier OPTIONAL,
        controlled [1] IMPLICIT SEQUENCE OF Identifier OPTIONAL
                -- this field shall appear if and only if the controlling field is included
        }
Select-Response ::= NULL
```

12.10.1 Select-Request (запрос выбора)

Абстрактный синтаксис выбора **Select** для запроса **ConfirmedServiceRequest** — это **Select-Request**.

12.10.2 Select-Response (ответ выбора)

Абстрактный синтаксис выбора **Select** для ответа **ConfirmedServiceResponse** — это **Select-Response**.

12.11 AlterProgramInvocationAttributes (изменение атрибутов вызова программы)

Описание абстрактного синтаксиса выбора **AlterProgramInvocationAttributes** для запроса **AdditionalService-Request** и ответа **AdditionalService-Response** определено типами **AlterProgramInvocationAttributes-Request** и **AlterProgramInvocationAttributes-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AlterProgramInvocationAttributes-Request ::= SEQUENCE {
    programInvocation                   [0] IMPLICIT Identifier,
    startCount                          [1] StartCount DEFAULT cycleCount: 1 }
AlterProgramInvocationAttributes-Response ::= NULL
```

12.11.1 AlterProgramInvocationAttributes-Request (запрос изменения атрибутов вызова программы)

Абстрактный синтаксис выбора **AlterProgramInvocationAttributes** для запроса **ConfirmedServiceRequest** — это **AlterProgramInvocationAttributes-Request**.

12.11.2 AlterProgramInvocationAttributes-Response (ответ изменения атрибутов вызова программы)

Абстрактный синтаксис выбора **AlterProgramInvocationAttributes** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AlterProgramInvocationAttributes-Response**.

12.12 ReconfigureProgramInvocation (активизация переконфигурации программы)

Описание абстрактного синтаксиса выбора **ReconfigureProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **ReconfigureProgramInvocation-Request** и **ReconfigureProgramInvocation-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReconfigureProgramInvocation-Request ::= SEQUENCE {
    oldProgramInvocationName        [0] IMPLICIT Identifier,
    newProgramInvocationName        [1] IMPLICIT Identifier OPTIONAL,
    domainsToAdd                    [2] IMPLICIT SEQUENCE OF Identifier,
    domainsToRemove                 [3] IMPLICIT SEQUENCE OF Identifier }
ReconfigureProgramInvocation-Response ::= NULL
```

12.12.1 ReconfigureProgramInvocation-Request (запрос активизации переконфигурации программы)

Абстрактный синтаксис выбора **ReconfigureProgramInvocation** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReconfigureProgramInvocation-Request**.

12.12.2 ReconfigureProgramInvocation-Response (ответ активизации переконфигурации программы)

Абстрактный синтаксис выбора **ReconfigureProgramInvocation** для запроса подтверждаемой услуги ответа подтверждаемой услуги **ConfirmedServiceResponse** – это **ReconfigureProgramInvocation-Response**.

13 Протокол управления блоком

13.1 Введение

В настоящем разделе приведено описание протокола реализации услуг, определенных в разделе 13 ИСО 9506-1. Элементы протокола:

InitiateUnitControlLoad	AddToUnitControl
UnitControlLoadSegment	RemoveFromUnitControl
UnitControlUpload	GetUnitControlAttributes
StartUnitControl	LoadUnitControlFromFile
StopUnitControl	StoreUnitControlToFile
CreateUnitControl	DeleteUnitControl

13.2 Элемент управления

Элемент управления — это комплексный параметр, используемый в нескольких услугах. Он представляет описание одного элемента блока управления объектом.

```
ControlElement ::= CHOICE {
    beginDomainDef          [0] SEQUENCE {
        domainName                  [1] IMPLICIT Identifier,
        capabilities                    [2] IMPLICIT SEQUENCE OF MMSString,
        sharable                        [3] IMPLICIT BOOLEAN,
        loadData                        [4] LoadData OPTIONAL
        },
    continueDomainDef           [1] SEQUENCE {
        domainName                  [1] IMPLICIT Identifier,
        loadData                        [3] LoadData
```

```
    },
endDomainDef            [2] IMPLICIT Identifier,
    piDefinition                    [3] IMPLICIT SEQUENCE {
    piName                          [0] IMPLICIT Identifier,
    listOfDomains               [1] IMPLICIT SEQUENCE OF Identifier,
    reusable                        [2] IMPLICIT BOOLEAN DEFAULT TRUE,
    monitorType                 [3] IMPLICIT BOOLEAN OPTIONAL,
    plState                         [4] IMPLICIT ProgramInvocationState OPTIONAL
    }   }
```

13.2.1 Monitor

Абстрактный синтаксис параметра **Monitor** элемента управления определен наличием или отсутствием поля **MonitorType**. Если поле **MonitorType** присутствует, то значение параметра **Monitor** равно **true**. Значение поля **MonitorType** должно указывать значение типа параметра **Monitor** запроса услуги в соответствии с 11.2.1.1.4 ИСО 9506-1.

13.3 Услуга InitiateUnitControlLoad (инициировать загрузку блока управления)

Описание абстрактного синтаксиса выбора **InitiateUCLoad** запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **InitiateUnitControlLoad-Request** и **InitiateUnitControlLoad-Response,** соответственно. Абстрактный синтаксис выбора **InitiateUCLoad** для ошибки дополнительной услуги **AdditionalService-Error** описан типом **InitiateUnitControl-Error**. Указанные типы приведены ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
InitiateUnitControlLoad-Request ::= Identifier -- Unit Control Name
InitiateUnitControlLoad-Response ::= NULL
InitiateUnitControl-Error ::= CHOICE {
    Domain                  [0] IMPLICIT Identifier,
    programInvocation       [1] IMPLICIT Identifier
    }
```

13.3.1 InitiateUnitControlLoad-Request (запрос инициирования загрузки блока управления)

Абстрактный синтаксис выбора **InitiateUCLoad** для типа **AdditionalService-Request** — это **InitiateUnitControlLoad-Request**.

13.3.2 InitiateUnitControlLoad-Response (ответ инициирования загрузки блока управления)

Абстрактный синтаксис выбора **InitiateUCLoad** для типа **AdditionalService-Response** — это **InitiateUnitControlLoad-Response**.

13.3.3 InitiateUnitControlLoad-Error (ошибка инициирования загрузки блока управления)

Абстрактный синтаксис выбора **InitiateUCLoad** для типа **AdditionalService-Error** — это **InitiateUnitControlLoad-Error**.

13.4 UnitControlLoadSegment (услуга сегмента загрузки блока управления)

Описание абстрактного синтаксиса выбора **UCLoad** для запроса **AdditionalService-Request** и ответа **AdditionalService-Response** определено типами **UnitControlLoadSegment-Request** и **UnitControlLoadSegment-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
UnitControlLoadSegment-Request ::= Identifier -- Unit Control Name
UnitControlLoadSegment-Response ::= SEQUENCE {
    controlElements             [0] IMPLICIT SEQUENCE OF ControlElement,
    moreFollows             [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

13.4.1 UnitControlLoadSegment-Request (запрос сегмента загрузки блока управления)

Абстрактный синтаксис выбора **UCLoad** для запроса дополнительной услуги **AdditionalService-Request** – это **UnitControlLoadSegment-Request.**

13.4.2 UnitControlLoadSegment-Response (ответ сегмента загрузки блока управления)

Абстрактный синтаксис выбора **UCLoad** для типа **AdditionalService-Response** — это **UnitControlLoadSegment-Response**.

13.5 UnitControlUpload (услуга подкачки блока управления)

Описание абстрактного синтаксиса выбора **UCUpload** запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **UnitControlUpload-Request** и **UnitControlUpload-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем разделе.

```
UnitControlUpload-Request ::= SEQUENCE {
    unitControlName             [0] IMPLICIT Identifier, -- Unit Control Name
    continueAfter               CHOICE {
        domain                      [1] IMPLICIT Identifier,
        ulsmID                      [2] IMPLICIT INTEGER,
        programInvocation           [3] IMPLICIT Identifier } OPTIONAL
    }
UnitControlUpload-Response ::= SEQUENCE {
    controlElements         [0] IMPLICIT SEQUENCE OF ControlElement,
    nextElement                 CHOICE {
        domain                      [1] IMPLICIT Identifier,
        ulsmID                      [2] IMPLICIT INTEGER,
        programInvocation           [3] IMPLICIT Identifier } OPTIONAL
     }
```

13.5.1 UnitControlUpload-Request (запрос подкачки блока управления)

Абстрактный синтаксис выбора **uCUpload** для типа **AdditionalService-Request** – это **UnitControlUpload-Request**.

13.5.2 UnitControlUpload-Response (ответ подкачки блока управления)

Абстрактный синтаксис выбора **UCUpload** для типа **AdditionalService-Response** — это **UnitControlUpload-Response**.

13.6 StartUnitControl (услуга запуска блока управления)

Описание абстрактного синтаксиса выбора **StartUC** для запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** задано типами **StartUnitControl-Request** и **StartUnitControl-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
StartUnitControl-Request ::= SEQUENCE {
    unitControlName         [0] IMPLICIT Identifier, -- Unit Control Name
    executionArgument           CHOICE {
        simpleString                [1] IMPLICIT MMSString,
        encodedString               EXTERNAL,
        enmbeddedString             EMBEDDED PDV } OPTIONAL
    }
StartUnitControl-Response ::= NULL
StartUnitControl-Error ::= SEQUENCE {
    programInvocationName       [0] IMPLICIT Identifier,
    programInvocationState      [1] IMPLICIT ProgramInvocationState }
```

13.6.1 StartUnitControl-Request (запрос запуска управления блоком)

Абстрактный синтаксис выбора **StartUC** для типа **AdditionalService-Request** — это **StartUnitControl-Request.**

13.6.2 StartUnitControl-Response (ответ запуска управления блоком)

Абстрактный синтаксис выбора **StartUC** для типа **AdditionalService-Response** — это **StartUnitControl-Response**.

13.6.3 StartUnitControl-Error (ошибка запуска управления блоком)

Абстрактный синтаксис выбора **StartUC** для типа **AdditionalService-Error** — это **StartUnitControl-Error.** Он является подпараметром имени вызова программы и подпараметром состояния вызова программы соответственно, для параметра **Result(–)** примитива ответа **StartUnitControl.response**. Он появляется как подпараметр имени вызова программы и подпараметр состояния вызова программы соответственно, для примитива подтверждения **StartUnitControl. confirm** (при его наличии).

13.7 Услуга StopUnitControl (останов управления блоком)

Описание абстрактного синтаксиса выбора **stopUC** для запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **StopUnitControl-Request** и **StopUnitControl-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
StopUnitControl-Request ::= Identifier -- Unit Control Name
StopUnitControl-Response ::= NULL
StopUnitControl-Error ::= SEQUENCE {
    programInvocationName       [0] IMPLICIT Identifier,
    programInvocationState      [1] IMPLICIT ProgramInvocationState }
```

13.7.1 StopUnitControl-Request (запрос останова управления блоком)

Абстрактный синтаксис выбора **StopUC** для типа **AdditionalService-Request** — это **StopUnitControl-Request**.

13.7.2 StopUnitControl-Response (ответ останова управления блоком)

Абстрактный синтаксис выбора **StopUC** для ответа дополнительной услуги **AdditionalService-Response** тип — это **StopUnitControl-Response**.

13.7.3 StopUnitControl-Error (ошибка останова управления блоком)

Абстрактный синтаксис выбора **StopUC** ошибки дополнительной услуги **AdditionalService-Error** — это **StopUnitControl-Error**. Он является подпараметром имени вызова программы и подпараметром состояния вызова программы соответственно, для параметра **Result(–)** примитива ответа **StopUnitControl.response.** Он имеет вид подпараметра имени вызова программы и подпараметра состояния вызова программы, соответственно, для примитива подтверждения **StopUnitControl.confirm** (при его наличии).

13.8 Услуга CreateUnitControl (создание блока управления)

Описание абстрактного синтаксиса выбора **CreateUC** запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **CreateUnitControl-Request** и **CreateUnitControl-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
CreateUnitControl-Request ::= SEQUENCE {
    unitControl                 [0] IMPLICIT Identifier, -- Unit Control Name
    domains                     [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations          [2] IMPLICIT SEQUENCE OF Identifier }
CreateUnitControl-Response ::= NULL
```

13.8.1 CreateUnitControl-Request (запрос создания блока управления)

Абстрактный синтаксис выбора **CreateUC** для типа **AdditionalService-Request** — это **CreateUnitControl-Request**.

13.8.2 CreateUnitControl-Response (ответ создания блока управления)

Абстрактный синтаксис выбора **CreateUC** для типа **AdditionalService-Response** — это **CreateUnitControl-Response**.

13.9 Услуга AddToUnitControl (добавить к управлению блоком)

Описание абстрактного синтаксиса выбора **AddToUC** запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **AddToUnitControl-Request** и **AddToUnitControl-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AddToUnitControl-Request ::= SEQUENCE {
    unitControl                 [0] IMPLICIT Identifier, -- Unit Control Name
    domains                     [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations          [2] IMPLICIT SEQUENCE OF Identifier }
AddToUnitControl-Response ::= NULL
```

13.9.1 AddToUnitControl-Request (запрос добавления к управлению блоком)

Абстрактный синтаксис выбора **AddToUC** для типа **AdditionalService-Request** — это **AddToUnitControl-Request**.

13.9.2 AddToUnitControl-Response (ответ добавления к управлению блоком)

Абстрактный синтаксис выбора **AddToUC** для типа **AdditionalService-Response** — это **AddToUnitControl-Response**.

13.10 Услуга RemoveFromUnitControl (удаление из блока управления)

Описание абстрактного синтаксиса выбора **RemoveFromUC** запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **RemoveFromUnitControl-Request** и **RemoveFromUnitControl-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
RemoveFromUnitControl-Request ::= SEQUENCE {
    unitControl                 [0] IMPLICIT Identifier, -- Unit Control Name
    domains                     [1] IMPLICIT SEQUENCE OF Identifier,
    programInvocations          [2] IMPLICIT SEQUENCE OF Identifier }
RemoveFromUnitControl-Response ::= NULL
```

13.10.1 RemoveFromUnitControl-Request (запрос удаления из блока управления)

Абстрактный синтаксис выбора **RemoveFromUC** для типа **AdditionalService-Request** — это **RemoveFromUnitControl-Request**.

13.10.2 RemoveFromUnitControl-Response (ответ удаления из блока управления)

Абстрактный синтаксис выбора **RemoveFromUC** для типа **AdditionalService-Response** — это **RemoveFromUnitControl-Response**.

13.11 Услуга GetUnitControlAttributes (получение атрибутов управления блоком)

Описание абстрактного синтаксиса выбора **GetUCAttributes** для запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **GetUnitControlAttributes-Request** и **GetUnitControlAttributes-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetUnitControlAttributes-Request ::= Identifier -- Unit Control Name
GetUnitControlAttributes-Response ::= SEQUENCE {
    domains                     [0] IMPLICIT SEQUENCE OF Identifier,
    programInvocations          [1] IMPLICIT SEQUENCE OF Identifier }
```

13.11.1 GetUnitControlAttributes-Request (запрос получения атибутов управления блоком)

Абстрактный синтаксис выбора **GetUCAttribute** для запроса дополнительной услуги **AdditionalService-Request** — это **GetUnitControlAttributes-Request**.

13.11.2 GetUnitControlAttributes-Response (ответ получения атибутов управления блоком)

Абстрактный синтаксис выбора **GetUCAttribute** для типа **AdditionalService-Response** — это **GetUnitControlAttributes-Response**.

13.12 Услуга LoadUnitControlFromFile (загрузка блока управления из файла)

Описание абстрактного синтаксиса выбора **LoadUCFromFile** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **LoadUnitControlFromFile-Request** и **LoadUnitControlFromFile-Response**, приведенными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    LoadUnitControlFromFile-Request ::= SEQUENCE {
        unitControlName                 [0] IMPLICIT Identifier,
        fileName                        [1] IMPLICIT FileName
IF ( tpy )
,       thirdParty                              [2] IMPLICIT ApplicationReference OPTIONAL
ENDIF
        }
    LoadUnitControlFromFile-Response ::= NULL
    LoadUnitControlFromFile-Error ::= CHOICE {
        none                            [0] IMPLICIT NULL,
        domain                          [1] IMPLICIT Identifier,
        programInvocation               [2] IMPLICIT Identifier
        }
```

13.12.1 LoadUnitControlFromFile-Request (запрос загрузки блока управления из файла)

Абстрактный синтаксис выбора **loadUCFromFile** для типа **AdditionalService-Request** — это **LoadUnitControlFromFile-Request**.

13.12.2 LoadUnitControlFromFile-Response (ответ загрузки блока управления из файла)

Абстрактный синтаксис выбора **loadUCFromFile** для типа **AdditionalService-Response** — это **LoadUnitControlFromFile-Response**.

13.12.3 LoadUnitControlFromFile-Error (ошибка загрузки блока управления из файла)

Абстрактный синтаксис выбора **loadUCFromFile** для типа **AdditionalService-Error** — это **LoadUnitControlFromFile-Error**.

13.13 Услуга StoreUnitControlToFile (хранение блока управления в файле)

Описание абстрактного синтаксиса выбора **StoreUCToFile** для запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** определено типами **StoreUnitControlToFile-Request** и **StoreUnitControlToFile-Response**, соответственно, приведенными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    StoreUnitControlToFile-Request ::= SEQUENCE {
        unitControlName                 [0] IMPLICIT Identifier,
        fileName                        [1] IMPLICIT FileName
IF ( tpy )
,       thirdParty                              [2] IMPLICIT ApplicationReference OPTIONAL
ENDIF
        }
    StoreUnitControlToFile-Response ::= NULL
```

13.13.1 StoreUnitControlToFile-Request (запрос хранения блока управления в файле)

Абстрактный синтаксис выбора **StoreUCToFile** для типа **AdditionalService-Request** — это **StoreUnitControlToFile-Request.**

13.13.2 StoreUnitControlToFile-Response (ответ хранения блока управления в файле)

Абстрактный синтаксис выбора **StoreUCToFile** для типа **AdditionalService-Response** — это **StoreUnitControlToFile-Response**.

13.14 Услуга DeleteUnitControl

Описание абстрактного синтаксиса выбора **DeleteUC** для запроса дополнительной услуги **AdditionalService-Request** и ответа дополнительной услуги **AdditionalService-Response** определено типами **DeleteUnitControl-Request** и **DeleteUnitControl-Response** соответственно, указанными ниже. В 5.5 устанавливает порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteUnitControl-Request ::= Identifier -- Unit Control Name
DeleteUnitControl-Response ::= NULL
DeleteUnitControl-Error ::= CHOICE   {
    domain                          [0] IMPLICIT Identifier,
    programInvocation               [1] IMPLICIT Identifier        }
```

13.14.1 DeleteUnitControl-Request (запрос удаления блока управления)

Абстрактный синтаксис выбора **DeleteUC** для типа **AdditionalService-Request** — это **DeleteUnitControl-Request**.

13.14.2 DeleteUnitControl-Response (ответ удаления блока управления)

Абстрактный синтаксис выбора **DeleteUC** для типа **AdditionalService-Response** — это **DeleteUnitControl-Response.**

13.14.3 DeleteUnitControl-Error (ошибка удаления блока управления)

Абстрактный синтаксис выбора **DeleteUC** для типа **AdditionalService-Error** — это **DeleteUnitControl-Error**.

14 Протокол доступа к переменной

В настоящем разделе приведено описание особых элементов протокола услуг, определенных обеспечением доступа к переменной для определения MMS-услуги. Данный раздел включает протокол, необходимый для реализации следующих услуг:

Read	DefineNamedVariableList
Write	GetNamedVariableListAttributes
InformationReport	DeleteNamedVariableLis
GetVariableAccessAttributes	DefineNamedType
DefineNamedVariable	GetNamedTypeAttributes
DeleteVariableAccess	DeleteNamedType

14.1 Соглашения

Все элементы протокола, рассмотренные в настоящем подразделе, удовлетворяют требованиям соглашений (см. 5.5), если не оговорено обратное. Всегда дается разъяснение в том случае, если указанные соглашения не выполнены в точности или когда имеется возможность их неоднозначной интерпретации. В добавление к соглашениям (см. 5.5) имеется соотношение между перенумерованным параметром определения услуги и протоколом, описанное ниже:

а) перенумерованный параметр примитива запроса (ответа), выбирающего между типами **CHOICE** (в перечне исключающих друг друга параметров), не должны указывать в протоколе. Рассматриваемый параметр должен иметь особое ссылочное имя, указывающее выбранный параметр. Данный выбранный параметр появляется в примитиве отображения (подтверждения, при его наличии) как перенумерованный параметр [со значением как в примитиве запроса (ответа)] или как выбранный параметр;

b) если значения перенумерованного параметра целые, то соответствие фактического значения параметра его значению в протоколе указано комментарием ASN.1.

Рассматриваемый параметр задан только одним из двух указанных определений.

14.2 Протокол спецификации типов

14.2.1 TypeSpecification (спецификация типа)

Абстрактный синтаксис параметра **TypeSpecification** описан ниже. В 14.1 содержится описание порядка получения всех параметров, не описанных явно в настоящем подразделе.

```
TypeSpecification ::= CHOICE    {
    typeName                    [0] ObjectName,
    typeDescription             TypeDescription }
```

14.3 Протокол спецификации альтернативного доступа

14.3.1 AlternateAccess (альтернативный доступ)

Абстрактный синтаксис параметра **AlternateAccess** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем пункте.

```
AlternateAccess ::= SEQUENCE OF CHOICE  {
    unnamed                                     AlternateAccessSelection
IF ( str2 )
,   named                                   [5] IMPLICIT SEQUENCE   {
      componentName                                 [0] IMPLICIT Identifier,
      access                                        AlternateAccessSelection    }
ENDIF
    }
AlternateAccessSelection ::= CHOICE {
    selectAlternateAccess                   [0] IMPLICIT SEQUENCE {
      accessSelection                               CHOICE {
IF ( str2 )
        component                           [0] IMPLICIT Identifier,
ELSE
      component                             [0] IMPLICIT NULL,
ENDIF
IF ( str1 )
        index                                       [1] IMPLICIT Unsigned32,
        indexRange                          [2] IMPLICIT SEQUENCE {
          lowIndex                                      [0] IMPLICIT Unsigned32,
          numberOfElements                      [1] IMPLICIT Unsigned32
          },
ELSE
        Index                                       [1] IMPLICIT NULL,
        indexRange                          [2] IMPLICIT NULL,
ENDIF
        allElements                         [3] IMPLICIT NULL
        },
    alternateAccess                 AlternateAccess
    },
  selectAccess                              CHOICE {
IF ( str2 )
      component                             [1] IMPLICIT Identifier,
ELSE
      component                             [1] IMPLICIT NULL,
ENDIF
IF ( str1 )
      index                                 [2] IMPLICIT Unsigned32,
      indexRange                            [3] IMPLICIT SEQUENCE {
```

```
            lowIndex                                    [0] IMPLICIT Unsigned32,
            numberOfElements                                [1] IMPLICIT Unsigned32
            },
ELSE
        index                                       [2] IMPLICIT NULL,
        indexRange                                  [3] IMPLICIT NULL,
ENDIF
        allElements                                 4] IMPLICIT NULL
        }       }
```

Тип **AlternateAccess** — это параметр **AlternateAccess**. Элементы переченя **AlternateAccessSelection** данного параметра должны содержаться в соответствующих элементах последовательного типа **AlternateAccess**. Каждый элемент должен участвовать в поименованном или непоименованном выборе, основанном на наличии или отсутствии, соответственно, параметра **ComponentName** для рассматриваемого элемента перечня **AlternateAccessSelection** данного параметра.

Если элемент перечня параметра **AlternateAccessSelection** содержит описание параметра **ComponentName**, то следует отдать предпочтение поименованному выбору. В данном случае **componentName** – это параметр **ComponentName,** а **access** – это тип **AlternateAccessSelection** описания особого выбора (см. далее).

Если элемент перечня для параметра **AlternateAccessSelection** не содержит описание параметра **ComponentName**, то следует отдать предпочтение непоименованному **unnamed** выбору. Это должен быть тип **AlternateAccessSelection,** он содержит описание особого выбора (см. далее).

Тип **AlternateAccessSelection** выводится из параметров (исключая **ComponentName**) соответствующего элемента перечня параметра **AlternateAccessSelection**. Порядок получения данного типа определен следующим образом:

a) если тип выбора содержит описание **SELECT-ALTERNATE-ACCESS**, то следует выбрать **SelectAlternateAccess**. Параметры настоящего выбора получены следующим образом:

1) поле **AccessSelection** получено из параметров **AccessSelection**, **Component**, **Index** и **IndexRange** в соответствии с 14.1. Если параметр **AccessSelection** содержит описание сущности **INDEX-RANGE** и если параметры **Low Index** и **Number of Elements** равны нулю, то необходимо выбрать **AllElements**, так как **AccessSelection** может быть выбран вместо **IndexRange** по желанию отправителя. Между двумя указанными альтернативами нет семантической разницы,

2) поле **AlternateAccess** получено из параметра **AlternateAccess** путем рекурсивной ссылки на указанную процедуру;

b) если рассматриваемый вид отбора содержит описание **SELECT-ACCESS**, то следует отдать предпочтение варианту **SelectAccess**. Поле **AccessSelection** получено из параметров **AccessSelection**, **Component**, **Index** и **IndexRange** в соответствии с 14.1, если параметр **AccessSelection** содержит описание **INDEX-RANGE** и если параметры **Low Index** и **Number Of Elements** оба равны нулю. Можно также выбрать **AllElements** для **AccessSelection** вместо **IndexRange** по желанию отправителя. Между указанными альтернативами нет семантической разницы.

14.4 Протокол спецификации значений данных

14.4.1 AccessResult

Абстрактный синтаксис параметра **AccessResult** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем пункте.

```
AccessResult ::= CHOICE {
    failure         [0] IMPLICIT DataAccessError,
    success   Data }
```

Поле **Success** указано параметром **Success** для параметра **AccessResult** по примитиву запроса (ответа) со значением **true.** Оно имеет вид параметра **Success** примитива отображения (подтверждения) со значением **true** (при его наличии).

Поле **Failure** указано параметром **Success** для параметра **AccessResult** в примитиве запроса (ответа) со значением **false**. Оно должно быть параметром **DataAccessError** для параметра **AccessResult** и представлено в виде параметра **Success** со значением **false** и параметра **DataAccessError** примитива отображения (подтверждения).

14.4.2 Data (данные)

Абстрактный синтаксис параметра **Data** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Data ::= CHOICE {
    -- context tag 0 is reserved for AccessResult
IF ( str1 )
    array                   [1] IMPLICIT SEQUENCE OF Data,
ELSE
    array                   [1] IMPLICIT NULL,
ENDIF
IF ( str2 )
    structure       [2] IMPLICIT SEQUENCE OF Data,
ELSE
    structure       [2] IMPLICIT NULL,
ENDIF
    boolean                 [3] IMPLICIT BOOLEAN,
    bit-string      [4] IMPLICIT BIT STRING,
    integer                 [5] IMPLICIT INTEGER,
    unsigned        [6] IMPLICIT INTEGER, -- shall not be negative
    floating-point          [7] IMPLICIT FloatingPoint,
    -- [8] is reserved
    octet-string            [9] IMPLICIT OCTET STRING,
    visible-string          [10] IMPLICIT VisibleString,
    generalized-time [11] IMPLICIT GeneralizedTime,
    binary-time             [12] IMPLICIT TimeOfDay,
    bcd                     [13] IMPLICIT INTEGER, -- shall not be negative
    booleanArray            [14] IMPLICIT BIT STRING,
    objId                   [15] IMPLICIT OBJECT IDENTIFIER,
    ...,
    mMSString [16] IMPLICIT MMSString
    }
```

14.4.2.1 Порядок получения

Порядок получения параметра **Data** представлен следующим образом:

a) если вид данных — это **ARRAY**, то следует выбрать **Array,** а контентом данной области должен быть параметр **List Of Data** для параметра **Array**. Элементы **List Of Data** расположены в поле **Array** в порядке, заданном перечнем параметра **List Of Data**.

Выбор **BooleanArray** (опция отправителя) в рассматриваемом представлении данных может быть использован вместо выбора **array,** когда элементы данных типа **Array** имеют также тип **Boolean**. В данном случае элементы параметра **List Of Data** массива **Array** (с нулевого до последнего элемента перечня) размещены в соответствующих перенумерованных битах массива **BooleanArray**. Значение **true** представлено единицей, значение **false** – нулем.

Семантических отличий между булевым массивом **BooleanArray** и просто массивом **Array,** содержащим значения типа **Boolean**, нет. Использование специального массива **BooleanArray** — это опция отправителя. Она выбирается для повышения эффективности передачи данных. Все получатели должны быть готовы к получению формы, отличной от **BooleanArray**;

b) если вид данных равен **STRUCTURE**, то следует выбирать сущность **Structure**. Контент данной области — это параметр **List Of Data** для параметра **Structure**. Элементы **List Of Data** указаны в области **Structure** в порядке, установленном параметром **List Of Data**;

c) если вид данных равен **SIMPLE**, то значение параметра **Class** задает выбор параметра **Data** в соответствии с 14.1.

14.4.2.2 Тип FloatingPoint (плавающая точка)

```
FloatingPoint ::= OCTET STRING
```

Тип **FloatingPoint** определяет простой тип со значениями, являющимися положительными и отрицательными действительными числами, включая нуль. Он включает также представления положительной и отрицательной бесконечности и сущность **NaN** (не число). Возможные значения, принимаемые сущностью **FloatingPoint**, могут быть ограниченными (см. далее).

Значения **FloatingPoint** включают знак *S*, значащую часть *M*, показатель степени E и ширину показателя степени N (N > 0). Значащая часть *M* — это число в нижеследующем диапазоне:

если E = 0, то:

0,0 <= *M* < 1,0

в противном случае:

1,0 <= *M* < 2,0.

Значение **FloatingPoint** содержит четыре части:

a) ширину показателя степени (число битов показателя степени);

b) знак (описывает знак **FloatingPoint)**;

c) показатель степени (значение показателя степени);

d) дробную часть (значение поля значащей части числа, лежащего справа от бинарной точки (в базовом представлении № 2).

Указанные четыре части сущности **FloatingPoint** представлены в октетной строке, содержащей два и более октетов. Первый октет указанной строки **OCTET STRING** содержит ширину показателя степени, представляемую бинарным целым. Оставшиеся части сущности **FloatingPoint** представлены в последующих октетах рассматриваемой октетной строки следующим образом:

a) биты последующих октетов нумеруются от нуля до «**k**». Нулевой бит — это наиболее значительный бит первого октета последовательности. Бит «**k**» — это наименее значительный бит последнего октета последовательности. Используя данную нумерацию, можно поставить в соответствие биты указанных выше частей значения **FloatingPoint** значениям последующих октетов следующим образом:

1) знак ставят в соответствие нулевому биту. Плюсу соответствует нуль, минусу – единица.

2) показателю степени ставят в соответствие (в порядке уменьшения значимости битов) биты от 1 до «**n**»,

3) дробной части ставят в соответствие (в порядке уменьшения значимости битов) биты от «**n + 1**» до «**k**».

Примечание — Для одного значения **FloatingPoint** возможны различные представления из-за различной возможной ширины показателя степени. Семантические отличия для различных представлений одного и того же значения **FloatingPoint** не указаны;

b) значение «**NaN**» представлено всеми значениями, имеющими показатель степени, содержащий все биты, равные единице, и дробную часть, содержащую по крайней мере один бит, равный единице. Значение знакового бита несущественно;

c) бесконечность представлено всеми значениями, имеющими показатель степени, содержащий все биты, равные единице, и дробную часть, содержащую все биты, равные нулю. Значение знакового бита определяет знак бесконечности;

d) значение нуль представлено всеми значениями, содержащими все биты показателя степени и дробную часть, равные нулю. Значение знакового бита несущественно;

e) ненулевое конечное число с плавающей точкой **FloatingPoint** представлено значением **FloatingPoint,** имеющим показатель степени, содержащий по крайней мере один бит, равный нулю. Значение представленной сущности **FloatingPoint** определено уравнением

V = −1*S* * F * 2**(2 − 2(N−1)) если E = 0;

V = −1*S* * (1+F) * 2**(E − 2(N−1) +1) в противном случае,

при этом значения членов настоящего уравнения определены следующим образом:

1) «**S**» — значение знакового бита,

2) «**E**» — значение показателя степени,

3) «**N**» — число битов в показателе степени,

4) «**F**» — сумма взвешенных значений битов дробной части.

Наиболее значительный бит дробной части (бит «n + 1» в нумерации, описанной в разделе 1) должен иметь взвешенное значение, равное значению данного бита, умноженному на 2^{-1}. Наименее значительный бит дробной части (бит «**k**» в нумерации, описанной в разделе 1) должен иметь взвешенное значение, равное значению данного бита, умноженному на $2^{(N-K)}$;

f) все прочие значения считаются некорректными.

Представление последующих октетов совместимо с представлением с плавающей точкой одинарной точности по IEEE 754. Здесь число последующих октетов равно четырем, а значение начального октета – 8. Совместимость с представлением с плавающей точкой двойной точности по IEEE 754 имеет место, если число последующих октетов равно восьми, а значение начального октета – 11.

Так как ИСО 9506-1 и настоящий стандарт допускают любое число битов в дробной и показательной частях представления значения с плавающей точкой (включая, и не только, указанные значения в формате по IEEE 754), и реальные системы могут не поддерживать все возможные значения рассматриваемых членов, то возникают ситуации, когда невозможно предоставить гарантии того, чтобы особые значения также представлялись в указанном виде. Если принимаемое значение с плавающей точкой невозможно представить в рамках рассматриваемой практической реализации, то действует нижеследующее правило:

a) если значение показателя степени представимо, а значение дроби непредставимо, то данная дробь округляется до ближайшего представимого значения, если наиболее значительный бит непредставимой части дроби содержит 1. В противном случае дробь обрезается;

b) если значение показателя степени непредставимо и все биты не равны 1, то:

1) если показатель степени отрицателен (показатель степени меньше смещения экспоненты), то реальное значение округляется до нуля. В противном случае,

2) реальное значение округляется до положительной (отрицательной) бесконечности в зависимости от знака значения с плавающей точкой;

c) если значение показателя степени непредставимо и все его биты равны единице, то:

1) если все биты дроби равны нулю, то следует использовать для реального значения представление с положительной (отрицательной) бесконечностью в соответствии со знаком значения с плавающей точкой. В противном случае,

2) для реального значения используется представление **NaN**.

14.4.2.3 Тип **BCD**

Для данных типа **BCD** значения передаются с помощью эквивалентных целых. Например, значение **BCD** равное 82 (т. е. '10000010'B) передается как целое 82 или как '1010010'B.

14.4.3 DataAccessError (ошибка доступа к данным)

Абстрактный синтаксис параметра **DataAccessError** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем пункте.

```
DataAccessError ::= INTEGER  {
    object-invalidated                  (0),
    hardware-fault                      (1),
    temporarily-unavailable     (2),
    object-access-denied                (3),
    object-undefined            (4),
    invalid-address                     (5),
    type-unsupported                    (6),
    type-inconsistent           (7),
    object-attribute-inconsistent       (8),
    object-access-unsupported           (9),
    object-non-existent                 (10),
    object-value-invalid                (11)
    } (0..11)
```

14.5 Протокол спецификации доступа к переменным

14.5.1 VariableAccessSpecification (спецификация доступа к переменной)

Абстрактный синтаксис параметра **VariableAccessSpecification** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем пункте.

```
VariableAccessSpecification ::= CHOICE {
    listOfVariable              [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification   VariableSpecification,
IF ( valt )
        alternateAccess                 [5] IMPLICIT AlternateAccess OPTIONAL
ENDIF
        }
IF ( vlis )
,       variableListName        [1] ObjectName
```

```
ENDIF
        }
```

14.5.2 VariableSpecification (спецификация переменной)

Абстрактный синтаксис параметра **VariableSpecification** задан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    VariableSpecification ::= CHOICE {
IF ( vnam )
        name                            [0] ObjectName,
ENDIF
IF ( vadr )
        address                 [1] Address,
        variableDescription     [2] IMPLICIT SEQUENCE {
                address                 Address,
                typeSpecification   TypeSpecification
                },
ENDIF
        -- the following element is only present to support the services
        -- defined in annex E
IF ( vsca )
        scatteredAccessDescription [3] IMPLICIT ScatteredAccessDescription,
ELSE
        scatteredAccessDescription [3] IMPLICIT NULL,
ENDIF
        invalidated [4] IMPLICIT NULL
        }
```

Выбор **Invalidated** определен параметром **KindOfVariable,** имеющим значение **INVALIDATED.** Он имеет вид параметра **Kind Of Variable**, имеющего значение **INVALIDATED**.

14.6 Read (читать)

Описание абстрактного синтаксиса выбора **Read** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** приведено ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    Read-Request ::= SEQUENCE {
        specificationWithResult         [0] IMPLICIT BOOLEAN DEFAULT FALSE,
        variableAccessSpecification     [1] VariableAccessSpecification }
    Read-Response ::= SEQUENCE {
        variableAccessSpecification     [0] VariableAccessSpecification OPTIONAL,
        listOfAccessResult                      [1] IMPLICIT SEQUENCE OF AccessResult }
```

14.6.1 Read-Request (запрос чтения)

Абстрактный синтаксис выбора **Read** для типа **ConfirmedServiceRequest** — это тип **Read-Request**.

14.6.2 Read-Response (ответ чтения)

Абстрактный синтаксис выбора **Read** для типа **ConfirmedServiceResponse** — это тип **Read-Response**.

14.7 Write (писать)

Абстрактный синтаксис выбора **Write** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    Write-Request ::= SEQUENCE {
        variableAccessSpecification                     VariableAccessSpecification,
        listOfData                                      [0] IMPLICIT SEQUENCE OF Data }
```

```
Write-Response ::= SEQUENCE OF CHOICE {
    failure                                    [0] IMPLICIT DataAccessError,
    success                                    [1] IMPLICIT NULL }
```

14.7.1 Write-Request (запрос записи)

Абстрактный синтаксис выбора **Write** для типа **ConfirmedServiceRequest** — это тип **Write-Request**.

14.7.2 Write-Response (ответ записи)

Абстрактный синтаксис выбора **Write** для типа **ConfirmedServiceResponse** — это тип **Write-Response**.

Поле **Success** указано параметром **Success** примитива ответа **Write.response**, имеющим значение **true**, и имеет вид параметра **Success**, имеющего значение **true**, для примитива подтверждения **Write.confirm**.

Поле **Failure** указано параметром **Success** примитива ответа **Write.response**, имеющим значение **false**. Оно должно быть параметром ошибки **DataAccessError** примитива ответа **Write.response** и иметь вид параметра **Success**, имеющего значение **false**, и параметра ошибки **DataAccessError** примитива подтверждения **Write.confirm**.

14.8 InformationReport (информационный отчет)

Абстрактный синтаксис выбора **InformationReport** неподтверждаемой услуги типа **UnconfirmedService** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
InformationReport ::= SEQUENCE {
    variableAccessSpecification        VariableAccessSpecification,
    listOfAccessResult                 [0] IMPLICIT SEQUENCE OF AccessResult }
```

14.8.1 InformationReport (информационный отчет)

Абстрактный синтаксис выбора **InformationReport** неподтверждаемой услуги типа **UnconfirmedService** — это тип **InformationReport**.

Примечание — Услуга **InformationReport** является неподтверждаемой.

14.9 GetVariableAccessAttributes (получение атрибутов доступа к переменной)

Абстрактный синтаксис выбора **GetVariableAccessAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    GetVariableAccessAttributes-Request ::= CHOICE {
IF ( vnam )
        name                           [0] ObjectName
IF ( vadr )
,
ENDIF
IF ( vadr)
        address                        [1] Address
ENDIF
        }
    GetVariableAccessAttributes-Response ::= SEQUENCE {
        mmsDeletable                   [0] IMPLICIT BOOLEAN,
IF ( vadr )
        address                        [1] Address OPTIONAL,
ENDIF
        typeDescription          [2] TypeDescription
IF ( aco )
,       accessControlList              [3] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
```

```
ENDIF
IF ( sem )
,       meaning                 [4] IMPLICIT VisibleString OPTIONAL
ENDIF
        }
```

14.9.1 GetVariableAccessAttributes-Request (запрос получения атрибутов доступа к переменной)

Абстрактный синтаксис выбора **GetVariableAccessAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **GetVariableAccessAttributes-Request**.

14.9.2 GetVariableAccessAttributes-Response (ответ получения атрибутов доступа к переменной)

Абстрактный синтаксис выбора **GetVariableAccessAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **GetVariableAccessAttributes-Response**.

14.9.2.1 AccessControlList (перечень средств управления доступом)

Параметр **AccessControlLis**t появляется только в том случае, если оговорен параметр **aco CBB**.

14.10 DefineNamedVariable (определение поименованной переменной)

Абстрактный синтаксис выбора **DefineNamedVariable** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineNamedVariable-Request ::= SEQUENCE {
    variableName                    [0] ObjectName,
    address                 [1] Address,
    typeSpecification       [2] TypeSpecification OPTIONAL }
DefineNamedVariable-Response ::= NULL
```

14.10.1 DefineNamedVariable-Request (запрос определения поименованной переменной)

Абстрактный синтаксис выбора **DefineNamedVariable** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DefineNamedVariable-Request**.

14.10.2 DefineNamedVariable-Response (ответ определения подтверждаемой услуги)

Абстрактный синтаксис выбора **DefineNamedVariable** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DefineNamedVariable-Response**, соответствующий типу **NULL**.

14.11 DeleteVariableAccess (удаление доступа к переменной)

Абстрактный синтаксис выбора **DeleteVariableAccess** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteVariableAccess-Request ::= SEQUENCE {
    scopeOfDelete               [0] IMPLICIT INTEGER {
        specific                        (0),
        aa-specific                 (1),
        domain                          (2),
        vmd                             (3)
        }   (0..3) DEFAULT specific,
    listOfName                  [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                  [2] IMPLICIT Identifier OPTIONAL }
DeleteVariableAccess-Response ::= SEQUENCE {
    numberMatched       [0] IMPLICIT Unsigned32,
    numberDeleted       [1] IMPLICIT Unsigned32 }
DeleteVariableAccess-Error ::= Unsigned32 -- numberDeleted
```

14.11.1 DeleteVariableAccess-Request (запрос удаления доступа к переменной)

Абстрактный синтаксис выбора **DeleteVariableAccess** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DeleteVariableAccess-Request**.

14.11.2 DeleteVariableAccess-Response (ответ удаления доступа к переменной)

Абстрактный синтаксис выбора **DeleteVariableAccess** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DeleteVariableAccess-Response**.

14.12 DefineNamedVariableList (определение списка поименованных переменных)

Абстрактный синтаксис выбора **defineNamedVariableList** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineNamedVariableList-Request ::= SEQUENCE {
    variableListName                    ObjectName,
    listOfVariable                      [0] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification           VariableSpecification
IF (valt)
,       alternateAccess                 [5] IMPLICIT AlternateAccess OPTIONAL
ENDIF
        }       }
DefineNamedVariableList-Response ::= NULL
```

14.12.1 DefineNamedVariableList-Request (запрос определения списка поименованных переменных)

Абстрактный синтаксис выбора **defineNamedVariableList** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DefineNamedVariableList-Request**.

14.12.2 DefineNamedVariableList-Response (ответ определения перечня поименованных переменных)

Абстрактный синтаксис выбора **defineNamedVariableList** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DefineNamedVariableList-Response**, соответствующий типу **NULL**.

14.13 GetNamedVariableListAttributes (получение атрибутов перечня поименованных переменных)

Абстрактный синтаксис выбора **GetNamedVariableListAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetNamedVariableListAttributes-Request ::= ObjectName – VariableListName
GetNamedVariableListAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    listOfVariable              [1] IMPLICIT SEQUENCE OF SEQUENCE {
        variableSpecification   VariableSpecification
IF ( valt )
,       alternateAccess                 [5] IMPLICIT AlternateAccess OPTIONAL }
ENDIF
IF ( aco )
,       accessControlList       [2] IMPLICIT Identifier OPTIONAL
            -- Shall not appear in minor version one or two
ENDIF
    }
```

14.13.1 GetNamedVariableListAttributes-Request (запрос получения атрибутов перечня поименованных переменных)

Абстрактный синтаксис выбора **GetNamedVariableListAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **GetNamedVariableListAttributes-Request**.

14.13.2 GetNamedVariableListAttributes-Response (ответ получения атрибутов перечня поименованных переменных)

Абстрактный синтаксис выбора **GetNamedVariableListAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **GetNamedVariableListAttributes-Response**.

14.13.2.1 AccessControlList (перечень средств управления доступом)

Параметр **AccessControlList** появляется только в том случае, если оговорен параметр **aco CBB**.

14.14 DeleteNamedVariableList (удаление перечня поименованных переменных)

Абстрактный синтаксис выбора **DeleteNamedVariableList** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteNamedVariableList-Request ::= SEQUENCE {
    scopeOfDelete                [0] IMPLICIT INTEGER {
        specific                         (0),
        aa-specific                      (1),
        domain                           (2),
        vmd                              (3)
        } (0..3) DEFAULT specific,
    listOfVariableListName   [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                   [2] IMPLICIT Identifier OPTIONAL }
DeleteNamedVariableList-Response ::= SEQUENCE {
    numberMatched            [0] IMPLICIT Unsigned32,
    numberDeleted            [1] IMPLICIT Unsigned32 }
DeleteNamedVariableList-Error ::= Unsigned32 -- numberDeleted
```

14.14.1 DeleteNamedVariableList-Request (запрос удаления перечня поименованных переменных)

Абстрактный синтаксис выбора **DeleteNamedVariableList** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DeleteNamedVariableList-Request**.

14.14.2 DeleteNamedVariableList-Response (ответ удаления перечня поименованных переменных)

Абстрактный синтаксис выбора **DeleteNamedVariableList** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DeleteNamedVariableList-Response**.

14.15 DefineNamedType (определение поименованного типа)

Абстрактный синтаксис выбора **defineNamedType** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineNamedType-Request ::= SEQUENCE {
    typeName                     ObjectName,
    typeSpecification        TypeSpecification }
DefineNamedType-Response ::= NULL
```

14.15.1 DefineNamedType-Request (запрос определения поименованного типа)

Абстрактный синтаксис выбора **defineNamedType** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DefineNamedType-Request**.

14.15.2 DefineNamedType-Response (ответ определения поименованного типа)

Абстрактный синтаксис выбора **defineNamedType** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DefineNamedType-Response**, соответствующий типу **NULL**.

14.16 GetNamedTypeAttributes (получение атрибутов поименованного типа)

Абстрактный синтаксис выбора **GetNamedTypeAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetNamedTypeAttributes-Request ::= ObjectName –TypeName
GetNamedTypeAttributes-Response ::= SEQUENCE {
    mmsDeletable                    [0] IMPLICIT BOOLEAN,
    typeSpecification               TypeSpecification
IF ( aco )
,   accessControlList               [1] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
ENDIF
IF ( sem )
,   meaning                         [4] IMPLICIT VisibleString OPTIONAL
    ENDIF
        }
```

14.16.1 GetNamedTypeAttributes-Request (запрос получения атрибутов поименованного типа)

Абстрактный синтаксис выбора **GetNamedTypeAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **GetNamedTypeAttributes-Request.**

14.16.2 GetNamedTypeAttributes-Response (ответ получения атрибутов поименованного типа)

Абстрактный синтаксис выбора **GetNamedTypeAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **GetNamedTypeAttributes-Response**.

14.16.2.1 AccessContrilList (перечень средств управления доступом)

Параметр **AccessControlList** появляется только в том случае, если оговорен параметр **aco CBB**.

14.17 DeleteNamedType (удаление поименованного типа)

Абстрактный синтаксис выбора **DeleteNamedType** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 14.1 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteNamedType-Request ::= SEQUENCE {
    scopeOfDelete               [0] IMPLICIT INTEGER {
        specific                    (0),
        aa-specific                 (1),
        domain                      (2),
        vmd                         (3)
        }   (0..3) DEFAULT specific,
    listOfTypeName          [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
    domainName                  [2] IMPLICIT Identifier OPTIONAL }
DeleteNamedType-Response ::= SEQUENCE {
    numberMatched           [0] IMPLICIT Unsigned32,
    numberDeleted           [1] IMPLICIT Unsigned32
    }
DeleteNamedType-Error ::= Unsigned32 -- numberDeleted
```

14.17.1 DeleteNamedType-Request (запрос удаления поименованного типа)

Абстрактный синтаксис выбора **DeleteNamedType** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DeleteNamedType-Request**.

14.17.2 DeleteNamedType-Response (ответ удаления поименованного типа)

Абстрактный синтаксис выбора **DeleteNamedType** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DeleteNamedType-Response**.

15 Протокол обмена данными

15.1 Введение

В настоящем разделе приведено описание протокола, необходимого для реализации услуг, определенных в разделе 15 ИСО 9506-1. Это услуги:

GetDataExchangeAttributes
ExchangeData

15.2 ExchangeData (данные для обмена)

Абстрактный синтаксис выбора **ExchangeData** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ExchangeData-Request ::= SEQUENCE {
    dataExchangeName                [0] ObjectName,
    listOfRequestData               [1] IMPLICIT SEQUENCE OF Data }
ExchangeData-Response ::= SEQUENCE {
    listOfResponseData              [0] IMPLICIT SEQUENCE OF Data }
```

15.2.1 ExchangeData-Request (запрос данных для обмена)

Абстрактный синтаксис выбора **ExchangeData** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ExchangeData-Request**.

15.2.2 ExchangeData-Response (ответ данных для обмена)

Абстрактный синтаксис выбора **ExchangeData** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ExchangeData-Response**.

15.3 GetDataExchangeAttributes (получение атрибутов данных для обмена)

Абстрактный синтаксис выбора **GetDataExchangeAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetDataExchangeAttributes-Request ::= ObjectName
GetDataExchangeAttributes-Response ::= SEQUENCE {
    inUse                               [0] IMPLICIT BOOLEAN,
    listOfRequestTypeDescriptions       [1] IMPLICIT SEQUENCE OF TypeDescription,
    listOfResponseTypeDescriptions      [2] IMPLICIT SEQUENCE OF TypeDescription,
    programInvocation                   [3] IMPLICIT Identifier OPTIONAL
IF (aco)
,   accessControlList                   [4] IMPLICIT Identifier OPTIONAL
ENDIF
                                -- Shall not appear in minor version one or two
    }
```

15.3.1 GetDataExchangeAttributes-Request (запрос получения атрибутов данных для обмена)

Абстрактный синтаксис выбора **GetDataExchangeAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetDataExchangeAttributes-Request**.

15.3.2 GetDataExchangeAttributes-Response (ответ получения атрибутов данных для обмена)

Абстрактный синтаксис выбора **GetDataExchangeAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetDataExchangeAttributes-Response.**

15.3.2.1 Активизация программы

Наличие элемент **ProgramInvocation** ответа **GetDataExchangeAttributes-Response** указывает на то, что значение атрибута **Linked** объекта обмена данными равно **true**. Если указанный элемент присутствует, то значение данного параметра должно доставлять имя вызова программы, на которую произведена ссылка ссылочным атрибутом активизации программы объекта обмена данными.

15.3.2.2 Перечень средств управления доступом

Параметр **AccessControlList** появляется только в том случае, если оговорен параметр **aco CBB.**

16 Протокол управления семафором

16.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуги управления семафором:

TakeControl
RelinquishControl
ReportSemaphoreStatus
ReportPoolSemaphoreStatus
DefineSemaphore
DeleteSemaphore
ReportSemaphoreEntryStatus

В добавление к указанным услугам настоящий раздел дает описания особых элементов протокола модификатора прикрепления к семафору **AttachToSemaphore**.

Все элементы протокола настоящего раздела соответствуют соглашению (см. 5.5), если не оговорено противное. Необходимые разъяснения даются, если указанные соглашения не применяю или когда существуют возможности неоднозначной интерпретации происходящего.

16.2 TakeControl (передача управления)

Абстрактный синтаксис выбора **TakeControl** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
TakeControl-Request ::= SEQUENCE {
    semaphoreName           [0] ObjectName,
    namedToken                  [1] IMPLICIT Identifier OPTIONAL,
    priority                    [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay         [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut          [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut          [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost  [6] IMPLICIT BOOLEAN DEFAULT TRUE
IF ( tpy )
,   applicationToPreempt    [7] IMPLICIT ApplicationReference OPTIONAL
ENDIF
    }
TakeControl-Response ::= CHOICE {
    noResult                [0] IMPLICIT NULL,
    namedToken                  [1] IMPLICIT Identifier }
```

16.2.1 TakeControl-Request (запрос передачи управления)

Абстрактный синтаксис выбора **TakeControl** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **TakeControl-Request**.

Поле **NamedToken** — это параметр **NamedToken** примитива ответа **TakeControl.response**. Он имеет вид параметра **NamedToken** примитива подтверждения **TakeControl.confirm**.

Если значение параметра приемлемой задержки в примитиве подтверждения является целым, то данное значение появляется как значение в поле **acceptableDelay**. Если значение параметра **AcceptableDelay** равно **FOREVER**, то поле **acceptableDelay** должно отсутствовать.

Если значение параметра **ControlTimeOut** примитива подтверждения является целым, то данное значение должно появляться как значение поля **ControlTimeOut**. Если значение параметра **ControlTimeOut** равно **FOREVER**, то поле **ControlTimeOut** должно отсутствовать.

Поле **abortOnTimeOut** присутствует, если и только если поле **AbortOnTimeOut** присутствует в примитиве подтверждения.

16.2.2 TakeControl-Response (ответ передачи управления)

Абстрактный синтаксис выбора **TakeControl** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **TakeControl-Response**.

16.3 RelinquishControl (отказ от управления)

Абстрактный синтаксис выбора **RelinquishControl** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
RelinquishControl-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    namedToken                 [1] IMPLICIT Identifier OPTIONAL }
RelinquishControl-Response ::= NULL
```

16.3.1 RelinquishControl-Request (запрос отказа от управления)

Абстрактный синтаксис выбора для **RelinquishControl** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **RelinquishControl-Request**.

16.3.2 RelinquishControl-Response (ответ отказа от управления)

Абстрактный синтаксис выбора для **RelinquishControl** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **RelinquishControl-Response**.

16.4 DefineSemaphore (определение семафора)

Абстрактный синтаксис выбора **defineСемафор** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineSemaphore-Request ::= SEQUENCE {
    semaphoreName              [0] ObjectName,
    numberOfTokens             [1] IMPLICIT Unsigned16 }
DefineSemaphore-Response ::= NULL
```

16.4.1 DefineSemaphore-Request (запрос определения семафора)

Абстрактный синтаксис выбора **defineСемафор** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineSemaphore-Request**.

16.4.2 DefineSemaphore-Response (ответ определения семафора)

Абстрактный синтаксис выбора **defineСемафор** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineSemaphore-Response**.

16.5 DeleteSemaphore (удаление семафора)

Абстрактный синтаксис выбора **DeleteSemaphore** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteSemaphore-Request ::= ObjectName -- Semaphore Name
DeleteSemaphore-Response ::= NULL
```

16.5.1 DeleteSemaphore-Request (запрос удаления семафора)

Абстрактный синтаксис выбора **DeleteSemaphore** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteSemaphore-Request.**

16.5.2 DeleteSemaphore-Response (ответ удаления семафора)

Абстрактный синтаксис выбора **DeleteSemaphore** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DeleteSemaphore-Response**.

16.6 ReportSemaphoreStatus (отчет о статусе семафора)

Абстрактный синтаксис выбора **ReportSemaphoreStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReportSemaphoreStatus-Request ::= ObjectName -- Semaphore Name
ReportSemaphoreStatus-Response ::= SEQUENCE {
```

```
        mmsDeletable                    [0] IMPLICIT BOOLEAN,
        class                           [1] IMPLICIT INTEGER {
                token                           (0),
                pool                            (1) } (0..1),
        numberOfTokens          [2] IMPLICIT Unsigned16,
        numberOfOwnedTokens     [3] IMPLICIT Unsigned16,
        numberOfHungTokens      [4] IMPLICIT Unsigned16
IF (aco)
,       accessControlList               [5] IMPLICIT Identifier OPTIONAL
                        -- Shall not appear in minor version one or two
ENDIF
        }
```

16.6.1 ReportSemaphoreStatus-Request (запрос отчета о статусе семафора)

Абстрактный синтаксис выбора **ReportSemaphoreStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportSemaphoreStatus-Request.**

16.6.2 ReportSemaphoreStatus-Response (ответ отчета о статусе семафора)

Абстрактный синтаксис выбора **ReportSemaphoreStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportSemaphoreStatus-Response.**

16.6.2.1 Перечень средств управления доступом

Параметр **AccessControlList** появляется только в том случае, если оговорен параметр **aco CBB**.

16.7 ReportPoolSemaphoreStatus (отчет о статусе семафора общего ресурса)

Абстрактный синтаксис выбора **ReportPoolSemaphoreStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    ReportPoolSemaphoreStatus-Request ::= SEQUENCE {
        semaphoreName                   [0] ObjectName,
        nameToStartAfter                        [1] IMPLICIT Identifier OPTIONAL }
    ReportPoolSemaphoreStatus-Response ::= SEQUENCE {
        listOfNamedTokens                       [0] IMPLICIT SEQUENCE OF CHOICE {
            freeNamedToken                              [0] IMPLICIT Identifier,
            ownedNamedToken                     [1] IMPLICIT Identifier,
            hungNamedToken                      [2] IMPLICIT Identifier },
        moreFollows                             [1] IMPLICIT BOOLEAN DEFAULT TRUE
        }
```

16.7.1 ReportPoolSemaphoreStatus-Request (запрос отчета о статусе семафора общего ресурса)

Абстрактный синтаксис выбора **ReportPoolSemaphoreStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportPoolSemaphoreStatus-Request**.

16.7.2 ReportPoolSemaphoreStatus-Response (ответ отчета остатусе семафора общего ресурса)

Абстрактный синтаксис выбора **ReportPoolSemaphoreStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportPoolSemaphoreStatus-Response**.

16.8 ReportSemaphoreEntryStatus (отчет о статусе записи семафора)

Абстрактный синтаксис выбора **ReportSemaphoreEntryStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    ReportSemaphoreEntryStatus-Request ::= SEQUENCE {
        semaphoreName           [0] ObjectName,
        state                           [1] IMPLICIT INTEGER {
            queued                              (0),
            owner                               (1),
```

```
        hung                                (2) } (0..2),
    entryIDToStartAfter                     [2] IMPLICIT OCTET STRING OPTIONAL }
ReportSemaphoreEntryStatus-Response ::= SEQUENCE {
    listOfSemaphoreEntry                    [0] IMPLICIT SEQUENCE OF SemaphoreEntry,
    moreFollows                             [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

16.8.1 ReportSemaphoreEntryStatus-Request (запрос отчета о статусе записи семафора)

Абстрактный синтаксис выбора **ReportSemaphoreEntryStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportSemaphoreEntryStatus-Reque**st.

16.8.2 ReportSemaphoreEntryStatus-Response (ответ отчета о статусе записи семафора)

Абстрактный синтаксис выбора **ReportSemaphoreEntryStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportSemaphoreEntryStatus-Response**.

16.8.3 SemaphoreEntry (запись семафора)

```
SemaphoreEntry ::= SEQUENCE {
    entryID                                 [0] IMPLICIT OCTET STRING,
    entryClass                              [1] IMPLICIT INTEGER {
        simple                                      (0),
        modifier                                    (1) } (0..1),
    applicationReference                    [2] ApplicationReference,
    namedToken                              [3] IMPLICIT Identifier OPTIONAL,
    priority                                [4] IMPLICIT Priority DEFAULT normalPriority,
    remainingTimeOut                        [5] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut                  [6] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost              [7] IMPLICIT BOOLEAN DEFAULT TRUE }
```

16.9 Модификатор AttachToSemaphore (прикрепление к семафору)

Абстрактный синтаксис выбора **AttachToSemaphore** типа модификатора описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AttachToSemaphore ::= SEQUENCE {
    semaphoreName               [0] ObjectName,
    namedToken                      [1] IMPLICIT Identifier OPTIONAL,
    priority                        [2] IMPLICIT Priority DEFAULT normalPriority,
    acceptableDelay             [3] IMPLICIT Unsigned32 OPTIONAL,
    controlTimeOut              [4] IMPLICIT Unsigned32 OPTIONAL,
    abortOnTimeOut              [5] IMPLICIT BOOLEAN OPTIONAL,
    relinquishIfConnectionLost      [6] IMPLICIT BOOLEAN DEFAULT TRUE }
```

17 Протокол связи с оператором

17.1 Введение

Настоящий раздел содержит описания блоков данных PDU для услуг связи с оператором. Настоящий раздел содержит описания протоколов, необходимых для реализации следующих услуг:

- Input
- Output

17.2 Input (вход)

Абстрактный синтаксис выбора **Input** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
Input-Request ::= SEQUENCE {
    operatorStationName         [0] IMPLICIT Identifier,
    echo                        [1] IMPLICIT BOOLEAN DEFAULT TRUE,
IF (output)
```

```
    listOfPromptData        [2] IMPLICIT SEQUENCE OF MMSString OPTIONAL,
ENDIF
    inputTimeOut                [3] IMPLICIT Unsigned32 OPTIONAL }
  Input-Response ::= MMSString -- Input String
```

17.2.1 Input-Request (запрос входа)

Абстрактный синтаксис выбора **Input** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **Input-Request**.

17.2.2 Input-Response (ответ входа)

Абстрактный синтаксис выбора **Input** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **Input-Response**, который является строкой **MMS String**.

17.3 Output (выход)

Абстрактный синтаксис выбора **Output** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
  Output-Request ::= SEQUENCE {
    operatorStationName             [0] IMPLICIT Identifier,
    listOfOutputData            [1] IMPLICIT SEQUENCE OF MMSString }
  Output-Response ::= NULL
```

17.3.1 Output-Request (запрос выхода)

Абстрактный синтаксис выбора **Output** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **Output-Request**.

17.3.2 Output-Response (ответ выхода)

Абстрактный синтаксис выбора **Output** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **Output-Response**.

18 Протокол управления событием

18.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуг и модификатора услуг, определенных функциональным блоком управления событием MMS. Раздел включает

```
  TriggerEvent                          GetAlarmSummary
  EventNotification                     GetAlarmEnrollmentSummary
  AcknowledgementEventNotification
```

18.2 TriggerEvent (событие запуска)

Абстрактный синтаксис выбора **TriggerEvent** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
  TriggerEvent-Request ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    priority                        [1] IMPLICIT Priority OPTIONAL }
  TriggerEvent-Response ::= NULL
```

18.2.1 TriggerEvent-Request (запрос события запуска)

Абстрактный синтаксис выбора **TriggerEvent** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **TriggerEvent-Request**.

18.2.2 TriggerEvent-Response (ответ события запуска)

Абстрактный синтаксис выбора **TriggerEvent** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **TriggerEvent-Response**.

18.3 EventNotification (уведомление о событии)

Абстрактный синтаксис выбора **eventNotification** типа **UnconfirmedService** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

Примечание — **EventNotification** — это неподтвержденная услуга. Поэтому она не определяет ответ или тип ошибки.

```
EventNotification ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
    eventConditionName                  [1] ObjectName,
    severity                            [2] IMPLICIT Severity,
    currentState                        [3] IMPLICIT EC-State OPTIONAL,
    TransitionTime                      [4] EventTime,
    notificationLost                    [6] IMPLICIT BOOLEAN DEFAULT FALSE,
    alarmAcknowledgmentRule             [7] IMPLICIT AlarmAckRule OPTIONAL,
    actionResult                        [8] IMPLICIT SEQUENCE {
        eventActionName                         ObjectName,
        successOrFailure                        CHOICE {
            success                                 [0] IMPLICIT SEQUENCE {
                confirmedServiceResponse                ConfirmedServiceResponse
IF ( csr cspi )
,               cs-Response-Detail [79] Response-Detail OPTIONAL
                                -- shall not be transmitted if value is the
                                -- value of a tagged type derived from NULL
ENDIF
                },
            failure                                 [1] IMPLICIT SEQUENCE {
                modifierPosition                        [0] IMPLICIT Unsigned32 OPTIONAL,
                serviceError                            [1] IMPLICIT ServiceError }
            }
        } OPTIONAL
    }
CS-EventNotification ::= [0] CHOICE {
IF ( des )
    string                              [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                               [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement               NULL    }
```

18.3.1 EventNotification (уведомление о событии)

Абстрактный синтаксис выбора **eventNotification** типа **UnconfirmedService** — это **EventNotification.** Порядок получения полей данного типа установлен ниже.

18.3.1.1 actionResult (результат действия)

Порядок получения поля **ActionResult** (при его наличии) соответствует 5.5. Если параметр **ActionResult** присутствует в примитиве запроса **EventNotification.request**, то его поле **successOrFailure** (успех или неудача) определено следующим образом:

a) если подпараметр **SuccessOrFailure** параметра **ActionResult** примитива запроса **EventNotification.request** равен **true**, то поле **SuccessOrFailure** должно содержать **Success,** а значение параметра **SuccessOrFailure** результата действия **ActionResult** примитива отображения **EventNotification.indication** (при его наличии) равно **true.** В противном случае поле **EventActionResult** должно содержать вариант **Failure,** а значение параметра **SuccessOrFailure** результата действия **ActionResult** примитива отображения **EventNotification.indication** (при его наличии) равно **false**;

b) если выбран вариант **Success,** то параметр **Result(+)** услуги, запрошенной полем **ConfirmedServiceRequest** объекта действия события, выражаем с помощью ответа подтверждаемой услуги **ConfirmedServiceResponse** выбора **Success** в соответствии с 5.5;

c) если выбран вариант **Failure** и отказ происходит при выполнени одного из модификаторов, описанных в поле модификаторов объекта действия события, то производится выбор **ModifierPosition** для варианта **Failure** с указанием модификатора, являющегося причиной отказа;

d) если выбран вариант **Failure** и отказ происходит при выполнении запрошенной подтверждаемой услуги, то параметр **Result(–)** услуги, запрошенной полем **ConfirmedServiceRequest** объекта действия события, выражаем с помощью выбора **ServiceError** для варианта **Failure** в соответствии с 5.5.

18.3.1.1.1 ConfirmedServiceResponse (ответ подтверждаемой услуги)

Абстрактный синтаксис параметра **ConfirmedServiceResponse** услуги **EventNotification** — это тип ответа подтверждаемой услуги **ConfirmedServiceResponse** с последующим выбором типа **CS-Response-Detail**, соответствующего выбору в отношении ответа подтверждаемой услуги **ConfirmedServiceResponse**.

18.3.1.2 Display Enhancement (расширение функциональности дисплея)

Абстрактный синтаксис выбора **EventNotification** неподтвержденных деталей **Unconfirmed-Detail** — это сущность **CS-EventNotification**, и данное поле должно доставлять параметр **DisplayEnhancement.**

18.4 AcknowledgementEventNotification (подтверждение уведомления о событии)

Абстрактный синтаксис выбора **AcknowledgementEventNotification** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AcknowledgeEventNotification-Request ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
    acknowledgedState                   [2] IMPLICIT EC-State,
    timeOfAcknowledgedTransition        [3] EventTime }
AcknowledgeEventNotification-Response ::= NULL
```

18.4.1 AcknowledgementEventNotification-Request (запрос подтверждения уведомления о событии)

Абстрактный синтаксис выбора **AcknowledgementEventNotification** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **AcknowledgementEventNotification-Request**.

18.4.2 AcknowledgementEventNotification-Response (ответ подтверждения уведомления о событии)

Абстрактный синтаксис выбора **AcknowledgementEventNotification** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AcknowledgementEventNotification-Response.**

18.5 GetAlarmSummary (получение заключения о сигнале опасности)

Абстрактный синтаксис выбора **GetAlarmSummary** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetAlarmSummary-Request ::= SEQUENCE {
    enrollmentsOnly             [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    activeAlarmsOnly                [1] IMPLICIT BOOLEAN DEFAULT TRUE,
    acknowledgementFilter       [2] IMPLICIT INTEGER {
        not-acked                       (0),
        acked                           (1),
        all                             (2)
    } (0..2) DEFAULT not-acked,
    severityFilter                  [3] IMPLICIT SEQUENCE {
        mostSevere                      [0] IMPLICIT Unsigned8,
        leastSevere                     [1] IMPLICIT Unsigned8 }
                                    DEFAULT { mostSevere 0, leastSevere 127 },
    continueAfter                   [5] ObjectName OPTIONAL
    }
GetAlarmSummary-Response ::= SEQUENCE {
    listOfAlarmSummary              [0] IMPLICIT SEQUENCE OF AlarmSummary,
```

```
        moreFollows                     [1] IMPLICIT BOOLEAN DEFAULT FALSE }
    AlarmSummary ::= SEQUENCE {
        eventConditionName              [0] ObjectName,
        severity                        [1] IMPLICIT Unsigned8,
        currentState                    [2] IMPLICIT EC-State,
        unacknowledgedState             [3] IMPLICIT INTEGER {
            none                            (0),
            active                          (1),
            idle                            (2),
            both                            (3)
            } (0..3),
IF ( csr cspi )
        displayEnhancement              [4] EN-Additional-Detail OPTIONAL,
                -- shall not be transmitted if the value is NULL
ELSE
        displayEnhancement              [4] NULL,
ENDIF
        timeOfLastTransitionToActive    [5] EventTime OPTIONAL,
        timeOfLastTransitionToIdle      [6] EventTime OPTIONAL }
    EN-Additional-Detail ::=            [0] CHOICE {
IF ( des )
        string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
        index                           [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement           NULL }
```

18.5.1 GetAlarmSummary-Request (запрос получения заключения о сигнале опасности)

Абстрактный синтаксис выбора **GetAlarmSummary** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetAlarmSummary-Request**.

18.5.2 GetAlarmSummary-Response (ответ получения заключения о сигнале опасности)

Абстрактный синтаксис выбора **GetAlarmSummary** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetAlarmSummary-Response**. Порядок формирования поля данного типа определен ниже.

18.5.2.1 ListOfAlarmSummary (перечень данных сигналов тревоги)

Поле **ListOfAlarmSummary** — это параметр **ListOfAlarmSummary** примитива ответа **GetAlarmSummary.response**. Данный параметр имеет вид параметра **ListOfAlarmSummary** примитива подтверждения **GetAlarmSummary.confirm**. Данное поле содержит нуль и более реализаций типа **AlarmSummary**, имеющих значение одного заключения **AlarmSummary**, описанного параметром **ListOfAlarmSummary**, полученным в установленном порядке.

18.5.2.1.1 displayEnhancement (увеличение функциональности дисплея)

Поле **displayEnhancement** заданного типа **AlarmSummary** — это параметр **DisplayEnhancement** для соответствующего **AlarmSummary** параметра **ListOfAlarmSummary** примитива ответа **GetAlarmSummary.response**. Данный параметр имеет вид параметра **DisplayEnhancement** соответствующего заключения **AlarmSummary** параметра **ListOfAlarmSummary** примитива подтверждения **GetAlarmSummary.confirm**. Абстрактный синтаксис данного поля — это тип **EN-Additional-Detail**, применяемый в соответствии с 5.5.

18.6 GetAlarmEnrollmentSummary (получение заключения о регистрации сигнала опасности)

Абстрактный синтаксис выбора **GetAlarmEnrollmentSummary** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
    GetAlarmEnrollmentSummary-Request ::= SEQUENCE {
        enrollmentsOnly             [0] IMPLICIT BOOLEAN DEFAULT TRUE,
```

```
        activeAlarmsOnly                    [1] IMPLICIT BOOLEAN DEFAULT TRUE,
        acknowledgementFilter               [2] IMPLICIT INTEGER {
            not-acked                               (0),
            acked                                   (1),
            all                                     (2)
        } (0..2) DEFAULT not-acked,
        severityFilter                      [3] IMPLICIT SEQUENCE {
            mostSevere                      [0] IMPLICIT Unsigned8,
            leastSevere                     [1] IMPLICIT Unsigned8 }
                                            DEFAULT { mostSevere 0, leastSevere 127 },
        continueAfter                       [5] ObjectName OPTIONAL
        }
    GetAlarmEnrollmentSummary-Response ::= SEQUENCE {
        listOfAlarmEnrollmentSummary    [0] IMPLICIT SEQUENCE OF AlarmEnrollmentSummary,
        moreFollows                         [1] IMPLICIT BOOLEAN DEFAULT FALSE }
    AlarmEnrollmentSummary ::= SEQUENCE {
        eventEnrollmentName                 [0] ObjectName,
IF ( tpy )
        clientApplication               [2] ApplicationReference OPTIONAL,
ELSE
        clientApplication               [2] NULL,
ENDIF
        severity                            [3] IMPLICIT Unsigned8,
        currentState                        [4] IMPLICIT EC-State,
IF ( cspi )
        displayEnhancement                  [5] EN-Additional-Detail OPTIONAL,
                        -- shall not be transmitted if the value is NULL
ELSE
        displayEnhancement                  [5] NULL,
ENDIF
        notificationLost                    [6] IMPLICIT BOOLEAN DEFAULT FALSE,
        alarmAcknowledgmentRule             [7] IMPLICIT AlarmAckRule,
        enrollmentState                 [8] IMPLICIT EE-State OPTIONAL,
        timeOfLastTransitionToActive        [9] EventTime OPTIONAL,
        timeActiveAcknowledged          [10] EventTime OPTIONAL,
        timeOfLastTransitionToIdle          [11] EventTime OPTIONAL,
        timeIdleAcknowledged                [12] EventTime OPTIONAL }
```

18.6.1 GetAlarmEnrollmentSummary-Request (запрос получения заключения о регистрации сигнала опасности)

Абстрактный синтаксис выбора **GetAlarmEnrollmentSummary** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetAlarmEnrollmentSummary-Request.**

18.6.2 GetAlarmEnrollmentSummary-Response (ответ получения заключения о регистрации сигнала опасности)

Абстрактный синтаксис выбора **GetAlarmEnrollmentSummary** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetAlarmEnrollmentSummary-Response**. Порядок формирования поля данного типа определен ниже.

18.6.2.1 ListOfAlarmEnrollmentSummary (перечень заключений о регистрации сигнала опасности)

Поле **ListOfAlarmEnrollmentSummary** — это параметр **ListOfAlarmEnrollmentSummary** примитива ответа **GetAlarmEnrollmentSummary.response**. Данный параметр имеет вид параметра **ListOfAlarmEnrollmentSummary** примитива подтверждения **GetAlarmEnrollmentSummary.confirm**. Данное поле содержит нуль и более реализаций типа **AlarmEnrollmentSummary** со значениями одного заключения **AlarmEnrollmentSummary**, описанного параметром **ListOfAlarmEnrollmentSummary**, полученным в установленном порядке.

18.6.2.1.1 displayEnhancement (повышение функциональности дисплея)

Поле **displayEnhancement** для заданного заключения **AlarmEnrollmentSummary** — это параметр **DisplayEnhancement** соответствующего заключения **AlarmEnrollmentSummary** для параметра перечня **ListOfAlarmEnrollmentSummary** примитива ответа **GetAlarmEnrollmentSummary. response**. Данный параметр имеет вид параметра **DisplayEnhancement** соответствующего заключения **AlarmEnrollmentSummary** в перечне **ListOfAlarmEnrollmentSummary** примитива подтверждения **GetAlarmEnrollmentSummary.confirm**. Абстрактный синтаксис данного поля — это тип **EN-AdditionalDetail**, соответствующий требованиям 5.5.

18.7 AttachToEventCondition (прикрепление к условию события)

Абстрактный синтаксис выбора **AttachToEventCondition** для рассматриваемого типа модификатора описан типом **AttachToEventCondition**, приведенным ниже ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AttachToEventCondition ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
    eventConditionName                  [1] ObjectName,
    causingTransitions                  [2] IMPLICIT Transitions,
    acceptableDelay             [3] IMPLICIT Unsigned32 OPTIONAL }
```

19 Протокол условий события

19.1 Введение

Настоящий раздел содержит описание особых элементов протокола услуг и модификатора услуг, определенных функциональным блоком управления событием MMS. Здесь рассмотрены услуги:

DefineEventCondition
DeleteEventCondition
GetEventConditionAttributes
ReportEventConditionStatus
AlterEventConditionMonitoring

19.2 DefineEventCondition (определение условий события)

Абстрактный синтаксис выбора **defineEventCondition** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineEventCondition-Request ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    class                           [1] IMPLICIT EC-Class,
    priority                        [2] IMPLICIT Priority DEFAULT normalPriority,
    severity                        [3] IMPLICIT Unsigned8 DEFAULT normalSeverity,
    alarmSummaryReports     [4] IMPLICIT BOOLEAN OPTIONAL,
    monitoredVariable               [6] VariableSpecification OPTIONAL,
    evaluationInterval              [7] IMPLICIT Unsigned32 OPTIONAL }
DefineEventCondition-Response ::= NULL
CS-DefineEventCondition-Request ::= [0] CHOICE {
IF ( des )
    string                          [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
    index                           [1] IMPLICIT INTEGER,
ENDIF
    noEnhancement NULL }
```

19.2.1 DefineEventCondition-Request (запрос определения условия события)

Абстрактный синтаксис выбора **defineEventCondition** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineEventCondition-Request**.

19.2.2 DefineEventCondition-Response (ответ определения условия события)

Абстрактный синтаксис выбора **defineEventCondition** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineEventCondition-Response**.

19.2.3 CS-DefineEventCondition-Request (запрос определения условия события типа CS)

Абстрактный синтаксис выбора **defineEventCondition** типа **Request-Detail** — это запрос **CS-DefineEventCondition-Request.** Данное поле доставляет значение параметра **DisplayEnhancement,** при его наличии.

19.3 DeleteEventCondition (удаление условия события)

Абстрактный синтаксис выбора **DeleteEventCondition** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteEventCondition-Request ::= CHOICE {
    specific                [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific             [1] IMPLICIT NULL,
    domain                  [2] IMPLICIT Identifier,
    vmd                     [3] IMPLICIT NULL }
DeleteEventCondition-Response ::= Unsigned32 --Candidates Not Deleted
```

19.3.1 DeleteEventCondition-Request (запрос удаления условия события)

Абстрактный синтаксис выбора **DeleteEventCondition** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteEventCondition-Request**. Значение данного выбора описано ниже.

Если значение параметра области применения удаления примитива услуги **DeleteEventCondition. request** равно **SPECIFIC**, то запрос **DeleteEventCondition-Request** содержит выбор **specific**. Данный выбор содержит значение параметра имени условия события примитива услуги **DeleteEventCondition. request**.

Если значение параметра области применения удаления примитива услуги **DeleteEventCondition. request** равно **AA-Specific**, то запрос **DeleteEventCondition-Request** содержит выбор **aa-specific**.

Если значение параметра области применения удаления примитива услуги **DeleteEventCondition. request** равно **DOMAIN**, то запрос **DeleteEventCondition-Request** содержит выбор **domain**. Данный выбор содержит значение параметра имени области примитива услуги **DeleteEventCondition.request**.

Если значение параметра области применения удаления примитива услуги **DeleteEventCondition. request** равно **VMD**, то запрос **DeleteEventCondition-Request** содержит выбор **vmd**.

19.3.2 DeleteEventCondition-Response (ответ удаления условия события)

Абстрактный синтаксис выбора **DeleteEventCondition** ответа подтверждаемой услуги **ConfirmedServiceResponse — это DeleteEventCondition-Response**. Здесь рассмотрен параметр нестираемых кандидатов примитива ответа **DeleteEventCondition.response**, указывающий значение **Result(+)**. Данный параметр имеет вид параметра нестираемых кандидатов примитива подтверждения **DeleteEventCondition.confirm** со значением **Result(+).**

19.4 GetEventConditionAttributes (получение атрибутов условия события)

Абстрактный синтаксис выбора **GetEventConditionAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetEventConditionAttributes-Request ::= ObjectName --Event Condition Name
GetEventConditionAttributes-Response ::= SEQUENCE {
    mmsDeletable            [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    class                   [1] IMPLICIT EC-Class,
    priority                [2] IMPLICIT Priority DEFAULT normalPriority,
    severity                [3] IMPLICIT Unsigned8 DEFAULT normalSeverity,
    alarmSummaryReports [4] IMPLICIT BOOLEAN DEFAULT FALSE,
    monitoredVariable       [6] CHOICE {
```

```
        variableReference           [0] VariableSpecification,
        undefined                       [1] IMPLICIT NULL } OPTIONAL,
    evaluationInterval              [7] IMPLICIT Unsigned32 OPTIONAL
IF (aco),
    accessControlList               [8] IMPLICIT Identifier OPTIONAL
ENDIF
                -- Shall not appear in minor version one or two
    }
  CS-GetEventConditionAttributes-Response ::= SEQUENCE {
    groupPriorityOverride           [0] CHOICE {
        priority                        [0] IMPLICIT Priority,
        undefined                       [1] IMPLICIT NULL } OPTIONAL,
        listOfReferencingECL  [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
        displayEnhancement    [2] CHOICE {
IF ( des )
        string                      [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
        index                       [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement                   [2] IMPLICIT NULL }
    }
```

19.4.1 GetEventConditionAttributes-Request (запрос получения атрибутов условия события)

Абстрактный синтаксис выбора **GetEventConditionAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** – это **GetEventConditionAttributes-Request.** Это параметр имени условия события примитива запроса **GetEventConditionAttributes.request**. Данный параметр имеет вид параметра имени условия события примитива отображения **GetEventConditionAttributes.indication** (при его наличии).

19.4.2 GetEventConditionAttributes-Response (ответ получения атрибутов условия события)

Абстрактный синтаксис выбора **GetEventConditionAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetEventConditionAttributes-Response**.

19.4.2.1 AlarmSummaryReports (отчет о сводке сигналов опасности)

Параметр имеет значение **false,** если значение параметра **Class** примитива услуги ответа **GetEventConditionAttributes.response** не равно **MONITORED**. В противном случае указанное значение равно значению параметра **AlarmSummaryReport** примитива услуги ответа **GetEventConditionAttributes.response**.

19.4.2.2 monitoredVariable (отслеживаемая переменная)

Поле **monitoredVariable** должно содержать выбор **Undefined,** если параметр **MonitoredVariable** примитива услуги ответа **GetEventConditionAttributes.response** имеет значение **UNDEFINED**. В противном случае выбирается вариант **VariableReference**.

19.4.2.3 AccessControlList (перечень органов управления доступом)

Параметр **AccessControlList** появляется в том случае, если и только если оговорено значение **aco CBB**.

19.4.3 CS-GetEventConditionAttributes-Response (ответ на получение атрибутов условия события типа CS)

Абстрактный синтаксис варианта **GetEventConditionAttributes** для подробностей ответа **Response-Detail** — это тип **CS-GetEventConditionAttributes-Response**.

19.5 ReportEventConditionStatus (статус отчета об условии события)

Абстрактный синтаксис выбора **ReportEventConditionStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReportEventConditionStatus-Request ::= ObjectName --Event Condition Name
ReportEventConditionStatus-Response ::= SEQUENCE {
    currentState                    [0] IMPLICIT EC-State,
    numberOfEventEnrollments        [1] IMPLICIT Unsigned32,
    enabled                         [2] IMPLICIT BOOLEAN OPTIONAL,
    timeOfLastTransitionToActive    [3] EventTime OPTIONAL,
    timeOfLastTransitionToIdle      [4] EventTime OPTIONAL }
```

19.5.1 ReportEventConditionStatus-Request (запрос статуса отчета об условии события)

Абстрактный синтаксис выбора **ReportEventConditionStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportEventConditionStatus-Request**.

19.5.2 ReportEventConditionStatus-Response (ответ статуса отчета об условии события)

Абстрактный синтаксис выбора **ReportEventConditionStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportEventConditionStatus-Response.**

19.6 AlterEventConditionMonitoring (мониторинг изменения условия события)

Абстрактный синтаксис выбора **AlterEventConditionMonitoring** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AlterEventConditionMonitoring-Request ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    enabled                         [1] IMPLICIT BOOLEAN OPTIONAL,
    priority                        [2] IMPLICIT Priority OPTIONAL,
    alarmSummaryReports         [3] IMPLICIT BOOLEAN OPTIONAL
IF ( cei )
,   evaluationInterval [4] IMPLICIT Unsigned32 OPTIONAL
ENDIF
                -- At least one of enabled, priority, alarmSummaryReports, or
                -- evaluationInterval shall be present.
    }
AlterEventConditionMonitoring-Response ::= NULL
CS-AlterEventConditionMonitoring-Request ::= SEQUENCE {
        changeDisplay CHOICE {
IF ( des )
        string                  [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
        index                   [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement                   [2] NULL } OPTIONAL
    }
```

19.6.1 AlterEventConditionMonitoring-Request (запрос мониторинга изменения условия события)

Абстрактный синтаксис выбора **AlterEventConditionMonitoring** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **AlterEventConditionMonitoring-Request**.

19.6.2 AlterEventConditionMonitoring-Response (ответ мониторинга изменения условия события)

Абстрактный синтаксис выбора **AlterEventConditionMonitoring** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AlterEventConditionMonitoring-Response**.

19.6.3 CS-AlterEventConditionMonitoring-Request (запрос мониторинга изменения условия события типа CS)

Абстрактный синтаксис выбора **AlterEventConditionMonitoring** детали запроса **Request-Detail** — это запрос **CS-AlterEventConditionMonitoring-Request**. Данное поле содержит параметр увели-

чения функциональности дисплея **DisplayEnhancement**. Данное поле используется в том случае, если и только если параметр **DisplayEnhancement** содержится в примитиве услуги отображения **AlterEventConditionMonitoring.indication.**

20 Протокол действия события

20.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуг и модификатора услуг, определяемых функциональным блоком управления событием MMS. Раздел содержит описания:

DefineEventAction — GetEventActionAttributes

DeleteEventAction — ReportEventActionStatus

20.2 DefineEventAction (определение действия события)

Абстрактный синтаксис выбора **defineEventAction** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineEventAction-Request ::= SEQUENCE {
    eventActionName             [0] ObjectName,
    listOfModifier              [1] IMPLICIT SEQUENCE OF Modifier OPTIONAL,
    confirmedServiceRequest     [2] ConfirmedServiceRequest
IF ( csr cspi )
,   cs-extension                [79] Request-Detail OPTIONAL
                    -- shall not be transmitted if value is the value
                    -- of a tagged type derived from NULL
ENDIF
    }
DefineEventAction-Response ::= NULL
```

20.2.1 DefineEventAction-Request (запрос определения действия события)

Абстрактный синтаксис выбора **defineEventAction** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineEventAction-Request.**

20.2.1.1 ConfirmedServiceRequest (запрос подтверждаемой услуги)

Абстрактный синтаксис выбора запроса подтверждаемой услуги параметр **EventAction** события услуг — это **ConfirmedServiceRequest** тип с последующим выбором детали запроса **CS-Request-Detail,** соответствующей выбору **ConfirmedServiceRequest**.

20.2.2 DefineEventAction-Response (ответ определения действия события)

Абстрактный синтаксис выбора **defineEventAction** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineEventAction-Response**.

20.3 DeleteEventAction (удаление действия события)

Абстрактный синтаксис выбора **DeleteEventAction** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteEventAction-Request ::= CHOICE {
    specific                    [0] IMPLICIT SEQUENCE OF ObjectName,
    aa-specific                 [1] IMPLICIT NULL,
    domain                      [3] IMPLICIT Identifier,
    vmd                         [4] IMPLICIT NULL }
DeleteEventAction-Response ::= Unsigned32 --Candidates Not Deleted
```

20.3.1 DeleteEventAction-Request (запрос тирания действия события)

Абстрактный синтаксис выбора **DeleteEventAction** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteEventAction-Request**. Значение данного выбора описано ниже.

Если значение параметра области применения удаления примитива услуги **DeleteEventAction. request** равно **SPECIFIC**, то запрос **DeleteEventAction-Request** содержит выбор **Specific.** Данный выбор содержит значение параметра имени действия события примитива услуги **DeleteEventAction.request**.

Если значение параметра области применения удаления примитива услуги **DeleteEventAction. request** равно **AA-Specific,** то запрос **DeleteEventAction-Request** содержит выбор **aa-specific**.

Если значение параметра области применения удаления примитива услуги **DeleteEventAction. request** равно **DOMAIN**, то запрос **DeleteEventAction-Request** содержит выбор **Domain**. Данный выбор содержит значение параметра имени области примитива услуги **DeleteEventAction.request**.

Если значение параметра области применения удаления примитива услуги **DeleteEventAction. request** равно **VMD**, то запрос **DeleteEventAction-Request** содержит выбор **vmd**.

20.3.2 DeleteEventAction-Response (ответ удаления действия события)

Абстрактный синтаксис выбора **DeleteEventAction** ответа подтверждаемой услуги **ConfirmedServiceResponse** – это **DeleteEventAction-Response**. Это должен быть параметр для «кандидатов на неудаление» примитива ответа **DeleteEventAction.response**, определяющего значение **Result(+)**. Данный параметр выглядит как параметр для «кандидатов на неудаление» примитива подтверждения **DeleteEventAction.confirm**, определяющего значение **Result(+).**

20.4 GetEventActionAttributes (получение атрибутов действия события)

Абстрактный синтаксис выбора **GetEventActionAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetEventActionAttributes-Request ::= ObjectName –EventActionName
GetEventActionAttributes-Response ::= SEQUENCE {
    mmsDeletable                    [0] IMPLICIT BOOLEAN DEFAULT FALSE,
    listOfModifier                  [1] IMPLICIT SEQUENCE OF Modifier,
    confirmedServiceRequest         [2] ConfirmedServiceRequest
IF ( csr cspi )
,   cs-extension                    [79] Request-Detail OPTIONAL
                    -- shall not be transmitted if value is the value
                    -- of a tagged type derived from NULL
ENDIF
IF (aco)
,   accessControlList [3] IMPLICIT Identifier OPTIONAL
ENDIF
                    -- Shall not appear in minor version one or two
    }
```

20.4.1 GetEventActionAttributes-Request (запрос на получение атрибутов действия события)

Абстрактный синтаксис выбора **GetEventActionAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetEventActionAttributes-Request.**

20.4.2 GetEventActionAttributes-Response (ответ на получение атрибутов действия события)

Абстрактный синтаксис выбора **GetEventActionAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetEventActionAttributes-Response**.

20.4.2.1 ConfirmedServiceRequest (запрос подтверждаемой услуги)

Абстрактный синтаксис выбора параметра запроса подтверждаемой услуги параметра получения атрибутов действия события — это тип **ConfirmedServiceRequest** с последующим выбором типа детали запроса **Request-Detail**, соответствующего выбору **ConfirmedServiceRequest**.

20.4.2.2 Перечень средств управления доступом

Параметр **AccessControlList** появляется в том случае, если и только если оговорено значение aco СВВ.

20.5 ReportEventActionStatus (статус отчета о действии события)

Абстрактный синтаксис выбора **ReportEventActionStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReportEventActionStatus-Request ::= ObjectName -- Event Action Name
ReportEventActionStatus-Response ::= Unsigned32 -- Number of Event Enrollments
```

20.5.1 ReportEventActionStatus-Request (запрос статуса отчета о действии события)

Абстрактный синтаксис выбора **ReportEventActionStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportEventActionStatus-Request**.

20.5.2 ReportEventActionStatus-Response (ответ статуса отчета о действии события)

Абстрактный синтаксис выбора **ReportEventActionStatus** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportEventActionStatus-Response**. Данный тип отображен величиной **Result(+),** содержащей № регистрации события в примитиве услуги ответа **ReportEventActionStatus. response.** Он выглядит как величина **Result(+),** содержащая номер регистрации события в примитиве подтверждения услуги **ReportEventActionStatus.confirm**.

21 Протокол регистрации события

21.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуг и модификатора услуг, определяемых функциональным блоком управления событием MMS. Раздел содержит описания услуг

DefineEventEnrollment
DeleteEventEnrollment
GetEventEnrollmentAttributes
ReportEventEnrollmentStatus
AlterEventEnrollment,

а также модификатора услуги прикрепления к условию события **AttachToEventCondition**.

21.2 DefineEventEnrollment (определение регистрации события)

Абстрактный синтаксис выбора **defineEventEnrollment** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
   DefineEventEnrollment-Request ::= SEQUENCE {
      eventEnrollmentName          [0] ObjectName,
      eventConditionName           [1] ObjectName,
      eventConditionTransitions         [2] IMPLICIT Transitions,
      alarmAcknowledgmentRule           [3] IMPLICIT AlarmAckRule,
      eventActionName              [4] ObjectName OPTIONAL
IF ( tpy )
,     clientApplication [5] ApplicationReference OPTIONAL
ENDIF
      }
   DefineEventEnrollment-Response ::= NULL
   DefineEventEnrollment-Error ::= ObjectName
   CS-DefineEventEnrollment-Request ::= [0] CHOICE {
IF ( des )
      string                            [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
      index                             [1] IMPLICIT INTEGER,
ENDIF
      noEnhancement                NULL }
```

21.2.1 DefineEventEnrollment-Request (запрос определения регистрации события)

Абстрактный синтаксис выбора **defineEventEnrollment** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineEventEnrollment-Request**.

21.2.2 DefineEventEnrollment-Response (ответ определения регистрации события)

Абстрактный синтаксис выбора **defineEventEnrollment** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineEventEnrollment-Response**.

21.2.3 DefineEventEnrollment-Error (ошибка определения регистрации события)

Абстрактный синтаксис выбора **defineEventEnrollment** для выбора **ServiceSpecificInformation** ошибки подтверждаемой услуги **ConfirmedServiceError —** это ошибка **DefineEventEnrollment-Error**, которая является параметром неопределенного объекта параметра **Result(–)** примитива ответа **DefineEventEnrollment.response**. Он выглядит как параметр неопределенного объекта параметра **Result(–)** примитива подтверждения **DefineEventEnrollment.confirm** (при его наличии).

21.2.4 CS-DefineEventEnrollment-Request (запрос определения регистрации объекта типа CS)

Абстрактный синтаксис выбора **defineEventEnrollment** для детали запроса **Request-Detail** — это **CS-DefineEventEnrollment-Request.** Он доставляет значение параметра повышения функциональности дисплея **DisplayEnhancement.**

21.3 DeleteEventEnrollment (удаление регистрации события)

Абстрактный синтаксис выбора **DeleteEventEnrollment** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DeleteEventEnrollment-Request ::= CHOICE {
    specific                    [0] IMPLICIT SEQUENCE OF ObjectName,
    ec                          [1] ObjectName,
    ea                          [2] ObjectName }
DeleteEventEnrollment-Response ::= Unsigned32 --Candidates Not Deleted
```

21.3.1 DeleteEventEnrollment-Request (запрос удаления регистрации события)

Абстрактный синтаксис выбора **DeleteEventCondition** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteEventEnrollment-Request**. Значение данного выбора описано ниже.

Если параметр области применения удаления отображает параметр перечня имен регистрации события, то для запроса **DeleteEventEnrollment-Request** следует выбрать величину **specific**. Данный выбор содержит значение параметра перечня имени регистрации события из примитива запроса **DeleteEventEnrollment.request**.

Если параметр области применения удаления отображает параметр имени условия события, то для запроса **DeleteEventEnrollment-Request** следует выбрать величину **ec**. Данный выбор содержит значение параметра имени условия события из примитива запроса **DeleteEventEnrollment.request**.

Если параметр области применения удаления отображает параметр имени действия события, то для запроса **DeleteEventEnrollment-Request** следует сделать выбор **ea**. Данный выбор содержит значение параметра имени действия события из примитива запроса **DeleteEventEnrollment.request**.

21.3.2 DeleteEventEnrollment-Response (ответ удаления регистрации события)

Абстрактный синтаксис выбора **DeleteEventEnrollment** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DeleteEventEnrollment-Response**. Это параметр «кандидатов на неудаление» из примитива ответа **DeleteEventEnrollment.response,** отображающего значение **Result(+)**. Он выглядит как параметр «кандидатов на неудаление» примитива подтверждения **DeleteEventEnrollment.confirm,** отображающего значение **Result(+)**.

21.4 GetEventEnrollmentAttributes (получение атрибутов регистрации события)

Абстрактный синтаксис выбора **GetEventEnrollmentAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
GetEventEnrollmentAttributes-Request ::= SEQUENCE {
    scopeOfRequest          [0] IMPLICIT INTEGER {
```

```
            specific                                (0),
            client                  (1),
            ec                                      (2),
            ea                                      (3)     }       (0..3) DEFAULT client,
        eventEnrollmentNames    [1] IMPLICIT SEQUENCE OF ObjectName OPTIONAL,
IF ( tpy )
        clientApplication       [2] ApplicationReference OPTIONAL,
ELSE
        clientApplication       [2] NULL,
ENDIF
        eventConditionName              [3] ObjectName OPTIONAL,
        eventActionName                 [4] ObjectName OPTIONAL,
        continueAfter                   [5] ObjectName OPTIONAL }
    GetEventEnrollmentAttributes-Response ::= SEQUENCE {
        listOfEEAttributes              [0] IMPLICIT SEQUENCE OF EEAttributes,
        moreFollows                     [1] IMPLICIT BOOLEAN DEFAULT FALSE }
    EEAttributes ::= SEQUENCE {
        eventEnrollmentName             [0] ObjectName,
        eventConditionName              [1] CHOICE {
            eventCondition                      [0] ObjectName,
            undefined                           [1] IMPLICIT NULL },
        eventActionName                 [2] CHOICE {
            eventAction                 [0] ObjectName,
            undefined                           [1] IMPLICIT NULL } OPTIONAL,
IF ( tpy )
        clientApplication       [3] ApplicationReference OPTIONAL,
ELSE
        clientApplication       [3] NULL,
ENDIF
        mmsDeletable                    [4] IMPLICIT BOOLEAN DEFAULT FALSE,
        enrollmentClass         [5] IMPLICIT EE-Class,
        duration                [6] IMPLICIT EE-Duration DEFAULT current,
        invokeID                [7] IMPLICIT Unsigned32 OPTIONAL,
        remainingAcceptableDelay        [8] IMPLICIT Unsigned32 OPTIONAL
IF ( csr cspi )
,       displayEnhancement              [9] CHOICE {
IF ( des )
            string                      [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
            index                       [1] IMPLICIT INTEGER,
ENDIF
            noEnhancement                       NULL }
                                -- shall not be transmitted if the value is NULL
ELSE
            displayEnhancement          [9] NULL
IF ( aco )
,       accessControlList               [11] IMPLICIT Identifier
                                -- shall not appear in minor version one or two
ENDIF
        }
```

21.4.1 GetEventEnrollmentAttributes-Request (запрос получения атрибутов регистрации события)

Абстрактный синтаксис выбора **GetEventEnrollmentAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetEventEnrollmentAttributes-Request.**

Поле **ContinueAfter** содержит идентификатор регистрации параметра **ContinueAfter** примитива запроса **GetEventEnrollmentAttributes.request**. Он выглядит как идентификатор регистрации параметра **ContinueAfter** примитива отображения **GetEventEnrollmentAttributes.indication** (при его наличии).

Если параметр **ContinueAfte**r отсутствует в примитиве запроса, то данное поле отсутствует в запросе подтверждаемой услуги **ConfirmedServiceRequest,** а параметр **ContinueAfter** отсутствует в примитиве отображения (при его наличии).

21.4.1.1 scopeOfRequest (область применения запроса)

Поле **ScopeOfRequest** отображает выбранное значение параметра области применения запроса для примитива запроса. Если в примитиве запроса выбран перечень имен регистрации события, то в поле **ScopeOfRequest** указывается вариант **specific**. Если в примитиве запроса выбран вариант «приложение клиента», то в поле **ScopeOfRequest** указывается вариант **client**. Если в примитиве запроса выбран вариант «имя условия события», то в поле **ScopeOfRequest** указывается вариант **ec**. Если в примитиве запроса выбран вариант «имя действия события», то в поле **ScopeOfRequest** указывается вариант **ea**.

21.4.1.2 EventEnrollmentNames (имена регистрации события)

Если для параметра области применения запроса выбран перечень **EventEnrollmentNames**, то данное поле должно содержать значение указанного параметра. В противном случае данное поле отсутствует.

21.4.1.3 ClientApplication (приложение клиента)

Если для параметра области применения запроса выбран вариант «приложение клиента» и данный параметр не описывает клиента MMS рассматриваемого запроса услуги, то указанное поле должно содержать значение указанного параметра. В противном случае данное поле отсутствует.

Если для параметра области применения запроса выбран вариант имени условия события (имени действия события) и опцией данного параметра является «приложение клиента», то данное поле должно содержать значение указанного параметра. В противном случае данное поле отсутствует.

21.4.1.4 EventConditionName (имя условия события)

Если для параметра области применения запроса выбран вариант имени условия события, то данное поле должно содержать значение указанного параметра. В противном случае данное поле отсутствует.

21.4.1.5 EventActionName (имя действия события)

Если для параметра области применения запроса выбран вариант имени действия события, то данное поле должно содержать значение данного параметра. В противном случае данное поле отсутствует.

21.4.2 GetEventEnrollmentAttributes-Response (ответ получения атрибутов регистрации события)

Абстрактный синтаксис выбора **GetEventEnrollmentAttributes** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetEventEnrollmentAttributes-Response**.

21.4.2.1 listOfEEAttributes (перечень атрибутов типа EE)

Поле **listOfEEAttributes** — это параметр **ListOfEEAttributes** примитива ответа **GetEventEnrollmentAttributes.response**. Он выглядит как параметр **ListOfEEAttributes** примитива подтверждения **GetEventEnrollmentAttributes.confirm**. Данное поле содержит нуль и более реализаций типа **EEAttributes**. Каждая реализация содержит значение одного параметра **EEAttributes** для параметра **ListOfEEAttributes**, взятого в указанном порядке. В 5.5 представлено применение к каждой реализации параметра **EEAttributes** для параметра **ListOfEEAttributes** с целью получения соответствующего элемента параметра **ListOfEEAttributes**.

21.4.2.1.1 EventConditionName (имя условия события)

Поле **EventConditionName** содержит выбор **undefined**, если параметр имени условия события примитива услуги ответа **GetEventEnrollmentAttributes.response** имеет значение **UNDEFINED**. В противном случае выбирается вариант **EventCondition**.

21.4.2.1.2 EventActionName (имя действия события)

Если данное поле включено, то для поля **EventActionName** выбирают вариант **undefined,** если параметр **EventActionName** примитива услуги ответа **GetEventEnrollmentAttributes.response** имеет значение **UNDEFINED**. В противном случае выбирают вариант **EventAction**.

21.4.2.1.3 Перечень средств управления доступом

Параметр **AccessControlList** появляется в том случае, если и только если оговорено значение **aco CBB.**

21.5 ReportEventEnrollmentStatus (статус отчета о регистрации события)

Абстрактный синтаксис выбора **ReportEventEnrollmentStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReportEventEnrollmentStatus-Request ::= ObjectName --Event Enrollment Name
ReportEventEnrollmentStatus-Response ::= SEQUENCE {
    eventConditionTransitions           [0] IMPLICIT Transitions,
    notificationLost                    [1] IMPLICIT BOOLEAN DEFAULT FALSE,
    duration                    [2] IMPLICIT EE-Duration,
    alarmAcknowledgmentRule             [3] IMPLICIT AlarmAckRule OPTIONAL,
    currentState                        [4] IMPLICIT EE-State }
```

21.5.1 ReportEventEnrollmentStatus-Request (запрос статуса отчета о регистрации события)

Абстрактный синтаксис выбора **ReportEventEnrollmentStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportEventEnrollmentStatus-Request**.

21.5.2 ReportEventEnrollmentStatus-Response (ответ статуса отчета о регистрации события)

Абстрактный синтаксис выбора **ReportEventEnrollmentStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportEventEnrollmentStatus-Response**.

21.6 AlterEventEnrollment (изменение регистрации события)

Абстрактный синтаксис выбора **AlterEventEnrollment** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AlterEventEnrollment-Request ::= SEQUENCE {
    eventEnrollmentName                 [0] ObjectName,
    eventConditionTransitions           [1] IMPLICIT Transitions OPTIONAL,
    alarmAcknowledgmentRule             [2] IMPLICIT AlarmAckRule OPTIONAL }
AlterEventEnrollment-Response ::= SEQUENCE {
    currentState                        [0] CHOICE {
        state                           [0] IMPLICIT EE-State,
        undefined                               [1] IMPLICIT NULL },
    TransitionTime                      [1] EventTime }
CS-AlterEventEnrollment-Request ::= SEQUENCE {
    changeDisplay           CHOICE {
IF ( des )
        string                      [0] IMPLICIT VisibleString,
ENDIF
IF ( dei )
        index                       [1] IMPLICIT INTEGER,
ENDIF
        noEnhancement                       [2] NULL } OPTIONAL }
```

21.6.1 AlterEventEnrollment-Request (запрос изменения регистрации события)

Абстрактный синтаксис выбора **AlterEventEnrollment** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **AlterEventEnrollment-Request**.

21.6.2 AlterEventEnrollment-Response (ответ изменения регистрации события)

Абстрактный синтаксис выбора **AlterEventEnrollment** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AlterEventEnrollment-Response**.

21.6.2.1 Текущее состояние

Поле **CurrentState** содержит выбор **undefined**, если значение параметра **CurrentState** примитива подтверждения **AlterEventEnrollment.confirm** равно **UNDEFINED**. В противном случае выбирают вариант **state**.

21.6.3 CS-AlterEventEnrollment-Request (запрос изменения регистрации события типа CS)

Абстрактный синтаксис выбора **AlterEventEnrollment** деталей запроса **Request-Detail** — это запрос **CS-AlterEventEnrollment-Request.** Он должен доставлять значение параметра **DisplayEnhancement**. Если параметр **DisplayEnhancement** присутствует в примитиве запроса, то поле **ChangeDisplay** появляется в поле запроса **CS-AlterEventEnrollment-Request** с выбранным соответствующим типом. Если параметр **DisplayEnhancement** не присутствует в примитиве запроса, то поле **changeDisplay** не указывают в поле запроса **CS-AlterEventEnrollment-Request.** При этом данное поле состоит из пустой последовательности **SEQUENCE**.

21.7 Поддержка разработки

Ниже рассмотрен абстрактный синтаксис различных определений поддержки типа для протокола управления событием.

21.7.1 EE-State

Тип **EE-State** используется некоторыми параметрами для представления информации о сложном состоянии объектов условия события и регистрации события.

```
EE-State ::= INTEGER {
    disabled (0),
    idle (1),
    active (2),
    activeNoAckA (3),
    idleNoAckI (4),
    idleNoAckA (5),
    idleAcked (6),
    activeAcked (7),
    undefined (8)
    }
```

22 Протокол перечня условий события

22.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуг и модификатора услуг, определяемых функциональным блоком перечня условий события MMS. Раздел содержит описания:

DefineEventConditionList
DeleteEventConditionList
AddEventConditionListReference
RemoveEventConditionListReference
GetEventConditionListAttribute
ReportEventConditionListStatus
AlterEventConditionListMonitoring

22.2 Протокол DefineEventConditionList (определение перечня условия события)

Абстрактный синтаксис выбора **DefineECL** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse,** соответственно, описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
DefineEventConditionList-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName,
    listOfEventConditionName            [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
,   listOfEventConditionListName        [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
        -- shall appear if an only if recl has been negotiated.
ENDIF
    }
DefineEventConditionList-Response ::= NULL
DefineEventConditionList-Error ::= ObjectName
```

22.2.1 DefineEventConditionList-Request (запрос определения перечня условия события)

Абстрактный синтаксис выбора **defineECL** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DefineEventConditionList-Request**.

22.2.2 DefineEventConditionList-Response (ответ определения перечня условия события)

Абстрактный синтаксис выбора **DefineECL** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DefineEventConditionList-Response.**

22.2.3 DefineEventConditionList-Error (ошибка определения перечня условия события)

Абстрактный синтаксис выбора **defineECL** для ошибки **AdditionalService-Error** — это ошибка определения **DefineEventConditionList-Error**, которая является объектом параметра ошибки для параметра **Result(–)** примитива ответа **DefineEventConditionList.response.** Данный объект выглядит как объект параметра ошибки примитива подтверждения **DefineEventConditionList.confirm** (при его наличии).

22.3 Протокол DeleteEventConditionList (удаления перечня условия события)

Абстрактный синтаксис выбора **DeleteECL** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse**, соответственно, описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

DeleteEventConditionList-Request ::= ObjectName – EventConditionListName
DeleteEventConditionList-Response ::= NULL

22.3.1 DeleteEventConditionList-Request (запрос удаления перечня условия события)

Абстрактный синтаксис выбора **DeleteECL** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteEventConditionList-Request**.

22.3.2 DeleteEventConditionList-Response (ответ удаления перечня условия события)

Абстрактный синтаксис выбора **DeleteECL** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DeleteEventConditionList-Response**.

22.4 Протокол AddEventConditionListReference (добавления ссылки на перечень условия события)

Абстрактный синтаксис выбора **AddECLReference** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AddEventConditionListReference-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName,
    listOfEventConditionName                [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
,   listOfEventConditionListName    [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
                -- shall appear if an only if recl has been negotiated.
ENDIF
    }
AddEventConditionListReference-Response ::= NULL
AddEventConditionListReference-Error ::= ObjectName
```

22.4.1 AddEventConditionListReference-Request (запрос добавления ссылки на перечень условия события)

Абстрактный синтаксис выбора **AddECLReference** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **AddEventConditionListReference-Request**.

22.4.2 AddEventConditionListReference-Response (ответ добавления ссылки на перечень условия события)

Абстрактный синтаксис выбора **AddECLReference** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AddEventConditionListReference-Response.**

22.4.3 AddEventConditionListReference-Error (ошибка добавления ссылки на перечень условия события)

Абстрактный синтаксис выбора **AddECLReference** ошибки **AdditionalService-Error** — это ошибка **AddEventConditionListReference-Error**, которая является объектом параметра ошибки для параметра **Result(–)** примитива ответа **AddEventConditionListReference.response**. Данный объект выглядит как объект параметра ошибки примитива подтверждения **AddEventConditionListReference.confirm** (при его наличии).

22.5 Протокол RemoveEventConditionListReference (удаления ссылки на перечень условия события)

Абстрактный синтаксис выбора **RemoveECLReference** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
   RemoveEventConditionListReference-Request ::= SEQUENCE {
      eventConditionListName                [0] ObjectName,
      listOfEventConditionName                    [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
, listOfEventConditionListName [2] IMPLICIT SEQUENCE OF ObjectName
-- shall appear if an only if recl has been negotiated.
ENDIF
      }
   RemoveEventConditionListReference-Response ::= NULL
   RemoveEventConditionListReference-Error ::= CHOICE {
      eventCondition                        [0] ObjectName,
      eventConditionList                          [1] ObjectName }
```

22.5.1 RemoveEventConditionListReference-Request (запрос удаления ссылки на перечень условия события)

Абстрактный синтаксис выбора **RemoveECLReference** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **RemoveEventConditionListReference-Request.**

22.5.2 RemoveEventConditionListReference-Response (ответ удаления ссылки на перечень условия события)

Абстрактный синтаксис выбора **RemoveECLReference** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **RemoveEventConditionListReference-Response.**

22.5.3 RemoveEventConditionListReference-Error (ошибка удаления ссылки на перечень условия события)

Абстрактный синтаксис выбора **RemoveECLReference** для ошибки **AdditionalService-Error** — это ошибка **RemoveEventConditionListReference-Error**, которая является объектом параметра ошибки **Result(–)** примитива ответа **RemoveEventConditionListReference.response**. Данный объект выглядит как объект параметра ошибки примитива подтверждения **RemoveEventConditionListReference. confirm** (при его наличии).

22.6 Протокол GetEventConditionListAttribute (получения атрибутов перечня условия события)

Абстрактный синтаксис выбора **GetECLAttribute** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
   GetEventConditionListAttributes-Request ::= ObjectName – eventConditionListName
   GetEventConditionListAttributes-Response ::= SEQUENCE {
      listOfEventConditionName                  [1] IMPLICIT SEQUENCE OF ObjectName
IF ( recl )
,     listOfEventConditionListName     [2] IMPLICIT SEQUENCE OF ObjectName OPTIONAL
                  -- shall appear if an only if recl has been negotiated.
ENDIF
      }
```

22.6.1 GetEventConditionListAttribute-Request (запрос получения атрибутов перечня условия события)

Абстрактный синтаксис выбора **GetECLAttribute** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **GetEventConditionListAttribute-Request**.

22.6.2 GetEventConditionListAttribute-Response (ответ получения атрибутов перечня условия события)

Абстрактный синтаксис выбора **GetECLAttribute** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **GetEventConditionListAttribute-Response.**

22.7 Протокол ReportEventConditionListStatus (отчета о статусе перечня условия события)

Абстрактный синтаксис выбора **ReportECLStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReportEventConditionListStatus-Request ::= SEQUENCE {
    eventConditionListName          [0] ObjectName, -- Event Condition List Name
    continueAfter                   [1] IMPLICIT Identifier OPTIONAL }
ReportEventConditionListStatus-Response ::= SEQUENCE {
    listOfEventConditionStatus      [1] IMPLICIT SEQUENCE OF EventConditionStatus,
    moreFollows                     [2] IMPLICIT BOOLEAN DEFAULT TRUE }
EventConditionStatus ::= SEQUENCE {
    eventConditionName              [0] ObjectName,
    currentState                    [1] IMPLICIT EC-State,
    numberOfEventEnrollments        [2] IMPLICIT Unsigned32,
    enabled                         [3] IMPLICIT BOOLEAN OPTIONAL,
    timeOfLastTransitionToActive    [4] EventTime OPTIONAL,
    timeOfLastTransitionToIdle      [5] EventTime OPTIONAL }
```

22.7.1 ReportEventConditionListStatus-Request (запрос отчета о статусе перечня условия события)

Абстрактный синтаксис выбора **ReportECLStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportEventConditionListStatus-Request**.

22.7.2 ReportEventConditionListStatus-Response (ответ отчета о статусе перечня условия события)

Абстрактный синтаксис выбора **ReportECLStatus** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportEventConditionListStatus-Response.**

22.8 Протокол AlterEventConditionListMonitoring (мониторинга изменения перечня условия события)

Абстрактный синтаксис выбора **alterECLMonitoring** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
AlterEventConditionListMonitoring-Request ::= SEQUENCE {
    eventConditionListName      [0] ObjectName,
    enabled                         [1] IMPLICIT BOOLEAN,
    priorityChange                  [2] CHOICE {
        priorityValue                   [0] IMPLICIT INTEGER,
        priorityReset                   [1] IMPLICIT NULL } OPTIONAL
    }
AlterEventConditionListMonitoring-Response ::= NULL
```

22.8.1 AlterEventConditionListMonitoring-Request (запрос мониторинга изменения перечня условия события)

Абстрактный синтаксис выбора **alterECLMonitoring** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **AlterEventConditionListMonitoring-Request**.

22.8.2 AlterEventConditionListMonitoring-Response (ответ мониторинга изменения перечня условия события)

Абстрактный синтаксис выбора **alterECLMonitoring** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **AlterEventConditionListMonitoring-Response.**

23 Протокол управления журналом

23.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуг, определяемых разделом управления журналом для услуг MMS. Раздел содержит описания следующих услуг:

ReadJournal
WriteJournal
InitializeJournal
ReportJournalStatus
CreateJournal
DeleteJournal

23.2 ReadJournal (читать журнал)

Абстрактный синтаксис выбора **ReadJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **ReadJournal-Request** и **ReadJournal-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
ReadJournal-Request ::= SEQUENCE {
    journalName                 [0] ObjectName,
    rangeStartSpecification     [1] CHOICE {
        startingTime                    [0] IMPLICIT TimeOfDay,
        startingEntry                   [1] IMPLICIT OCTET STRING } OPTIONAL,
    rangeStopSpecification      [2] CHOICE {
        endingTime                      [0] IMPLICIT TimeOfDay,
        numberOfEntries                 [1] IMPLICIT Integer32 } OPTIONAL,
    listOfVariables             [4] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
    entryToStartAfter           [5] IMPLICIT SEQUENCE {
        timeSpecification               [0] IMPLICIT TimeOfDay,
        entrySpecification              [1] IMPLICIT OCTET STRING } OPTIONAL
    }
ReadJournal-Response ::= SEQUENCE {
    listOfJournalEntry          [0] IMPLICIT SEQUENCE OF JournalEntry,
        moreFollows             [1] IMPLICIT BOOLEAN DEFAULT FALSE }
JournalEntry ::= SEQUENCE {
    entryIdentifier             [0] IMPLICIT OCTET STRING,
    originatingApplication      [1] ApplicationReference,
    entryContent                [2] IMPLICIT EntryContent }
```

23.2.1 ReadJournal-Request (запрос чтения журнала)

Абстрактный синтаксис выбора **ReadJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReadJournal-Request**.

23.2.2 ReadJournal-Response (ответ чтения журнала)

Абстрактный синтаксис выбора **ReadJournal** ответа подтверждаемой услуги ConfirmedServiceResponse — это **ReadJournal-Response.**

23.3 WriteJournal (делать записи в журнале)

Абстрактный синтаксис выбора **WriteJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **WriteJournal-Request** и **WriteJournal-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
WriteJournal-Request ::= SEQUENCE {
    journalName                 [0] ObjectName,
```

```
    listOfJournalEntry                  [1] IMPLICIT SEQUENCE OF EntryContent }
  WriteJournal-Response ::= NULL
```

23.3.1 WriteJournal-Request (запрос записи в журнале)

Абстрактный синтаксис выбора **WriteJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **WriteJournal-Request**.

23.3.2 WriteJournal-Response (ответ записи в журнале)

Абстрактный синтаксис выбора **WriteJournal** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **WriteJournal-Response**.

23.4 InitializeJournal (инициализировать журнал)

Абстрактный синтаксис выбора **InitializeJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **InitializeJournal-Request** и **InitializeJournal-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
  InitializeJournal-Request ::= SEQUENCE {
    journalName             [0] ObjectName,
    limitSpecification  [1] IMPLICIT SEQUENCE {
      limitingTime                  [0] IMPLICIT TimeOfDay,
      limitingEntry                 [1] IMPLICIT OCTET STRING OPTIONAL } OPTIONAL
    }
  InitializeJournal-Response ::= Unsigned32 -- Entries Deleted
```

23.4.1 InitializeJournal-Request (запрос инициализации журнала)

Абстрактный синтаксис выбора **InitializeJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **InitializeJournal-Request**.

23.4.2 InitializeJournal-Response (ответ инициализации журнала)

Абстрактный синтаксис выбора **InitializeJournal** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **InitializeJournal-Response**.

Данное поле — это параметр удаления записей **EntriesDeleted** примитива ответа **InitializeJournal. response.** Он выглядит как параметр **EntriesDeleted** примитива подтверждения **InitializeJournal. confirm** (при его наличии).

23.5 ReportJournalStatus (отчет о статусе журнала)

Абстрактный синтаксис выбора **ReportJournalStatus** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **ReportJournalStatus-Request** и **ReportJournalStatus-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
  ReportJournalStatus-Request ::= ObjectName --Journal Name
  ReportJournalStatus-Response ::= SEQUENCE {
    currentEntries              [0] IMPLICIT Unsigned32,
    mmsDeletable                [1] IMPLICIT BOOLEAN
IF ( aco )
,   accessControlList           [2] IMPLICIT Identifier OPTIONAL
                -- Shall not appear in minor version one or two
ENDIF
    }
```

23.5.1 ReportJournalStatus-Request (запрос отчета о статусе журнала)

Абстрактный синтаксис выбора **ReportJournalStatus** для запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ReportJournalStatus-Request**.

Данное поле — это параметр имени журнала для примитива запроса **ReportJournalStatus. request.** Данный параметр выглядит как параметр имени журнала для примитива отображения **ReportJournalStatus.indication** (при его наличии).

23.5.2 ReportJournalStatus-Response (ответ отчета о статусе журнала)

Абстрактный синтаксис выбора **ReportJournalStatus** для ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **ReportJournalStatus-Response**.

23.5.2.1 Перечень средств управления доступом

Параметр **AccessControlList** появляется в том случае, если и только если оговорено значение **aco CBB.**

23.6 CreateJournal (создать журнал)

Абстрактный синтаксис выбора **CreateJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **CreateJournal-Request** и **CreateJournal-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем подразделе.

```
CreateJournal-Request ::= SEQUENCE {
    journalName                 [0] ObjectName }
CreateJournal-Response ::= NULL
```

23.6.1 CreateJournal-Request (запрос создания журнала)

Абстрактный синтаксис выбора **CreateJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **CreateJournal-Request**.

23.6.2 CreateJournal-Response (ответ создания журнала)

Абстрактный синтаксис выбора **CreateJournal** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **CreateJournal-Response**.

23.7 DeleteJournal (стереть журнал)

Абстрактный синтаксис выбора **DeleteJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан типами **DeleteJournal-Request** и **DeleteJournal-Response** соответственно, указанными ниже. В 5.5 установлен порядок получения всех параметров, не описанных явно в настоящем разделе.

```
DeleteJournal-Request ::= SEQUENCE {
    journalName                 [0] ObjectName }
DeleteJournal-Response ::= NULL
```

23.7.1 DeleteJournal-Request (запрос удаления журнала)

Абстрактный синтаксис выбора **DeleteJournal** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **DeleteJournal-Request**.

23.7.2 DeleteJournal-Response (ответ удаления журнала)

Абстрактный синтаксис выбора **DeleteJournal** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **DeleteJournal-Response**.

23.8 Поддержка разработки

23.8.1 EntryContent (контент записи)

```
EntryContent ::= SEQUENCE {
    occurrenceTime          [0] IMPLICIT TimeOfDay,
    entryForm                   CHOICE {
        data                        [2] IMPLICIT SEQUENCE {
            event                       [0] IMPLICIT SEQUENCE {
                eventConditionName                  [0] ObjectName,
                currentState                        [1] IMPLICIT EC-State } OPTIONAL,
        listOfVariables             [1] IMPLICIT SEQUENCE OF Journal-Variable OPTIONAL
        },
    annotation                      [3] MMSString }
    }
```

23.8.1.1 EntryForm (форма записи)

В запросе записи в журнале **WriteJournal-Request** делается выбор **data**, если значение параметра **EntryForm** примитива запроса **WriteJournal.request** равно **DATA**. В запросе записи **WriteJournal-Request** делается выбор **annotation,** если значение параметра **EntryForm** примитива запроса **WriteJournal.request** равно **ANNOTATION**.

В ответе **ReadJournal-Response** делается выбор **data**, если значение параметра **EntryForm** примитива ответа **ReadJournal.response** равно **DATA**. В ответе **ReadJournal-Response** делается выбор **annotation**, если значение параметра **EntryForm** примитива ответа **ReadJournal.response** равно **ANNOTATION**.

Нижеследующее утверждение **END** завершает модуль, открытый в разделе 7.

END

24 Отображение на нижележащие услуги связи

Настоящий раздел определяет алгоритм, по которому услуги, обеспечиваемые нижележащими системами связи, используются механизмом разработки протокола обмена сообщениями (ММРМ). Любое использование указанных услуг, отличное от описанного в настоящем разделе, приводит к ошибке протокола.

Протокол MMS (спецификации производственных сообщений) располагается в среде взаимосвязи открытых систем внутри прикладного уровня. Как прикладной сервисный элемент (ASE), настоящий стандарт использует (отображается на) услуги и примитивы услуг нижележащей системы связи. MMS-пользователи могут также оказаться элементами внутри рассматриваемого прикладного процесса, а также внутри других элементов услуг приложений ASE.

24.1 Отображения PDU (блоков данных протокола)

Все MMS PDU (блоки данных протокола спецификации производственных сообщений) рассматриваются как данные пользователя на нижележащем примитиве услуг. Отображение блока данных протокола PDU на услуги произведено следующим образом [все PDU отсылаются на примитив запроса (ответа) и получаются на примитив отображения (подтверждения) услуги]:

MMS PDU	Underlying Communication Service Primitive
Confirmed-RequestPDU	M-DATA request, indication
Confirmed-ResponsePDU	M-DATA request, indication
Confirmed-ErrorPDU	M-DATA request, indication
Unconfirmed-RequestPDU	M-DATA request, indication
RejectPDU	M-DATA request, indication
Cancel-RequestPDU	M-DATA request, indication
Cancel-ResponsePDU	M-DATA request, indication
Cancel-ErrorPDU	M-DATA request, indication
Initiate-RequestPDU	M-ASSOCIATE request, indication
Initiate-ResponsePDU	M-ASSOCIATE response, confirm (with Result parameter accepted)
Initiate-ErrorPDU	M-ASSOCIATE response, confirm (with Result parameter rejected)
Conclude-RequestPDU	M-RELEASE request, indication
Conclude-ResponsePDU	M-RELEASE response, confirm (with Result parameter accepted)
Conclude-ErrorPDU	M-RELEASE response, confirm (with Result parameter rejected)

Все прочие отображения MMS PDU на указанные услуги — это ошибки протокола.

24.2 Данные типа M-ASSOCIATE

Данные запроса (ответа) типа **M-ASSOCIATE** используются для инициализации атрибутов объектов прикладной ассоциации, созданных для данной ассоциации. Параметр ссылки приложения Application Reference (далее – ссылка приложения) используется для идентификации одноранговых

MMS-пользователей в рассматриваемой ассоциации. Данное значение идентифицирует: (1) узел коммуникации, (2) пользовательский процесс внутри данного узла. Данное значение использовано для установления значения клиентского поля объекта **Application-Association** (см. 8.2). Для вызова MMS-пользователя применен параметр отвечающая ссылка приложения (Responding Application Reference) как значение клиентского поля. Для вызванного MMS-пользователя поле вызова ссылки приложения используется как значение клиентского поля. Если значение аутентификации присутствует в примитиве запроса (ответа), то значение данного параметра должно использоваться для инициализации поля значения аутентификации объекта **Application-Association**.

24.3 Прекращение прикладной ассоциации

После получения корректного блока данных **Conclude-RequestPDU** механизм MMPM выдает примитив запроса **M-RELEASE.request** с блоком данных **Conclude-RequestPDU** пользователя.

После получения корректного блока данных **Conclude-ResponsePDU** с результирующим параметром, отображающим успешный выпуск прикладной ассоциации, механизм MMPM доставляет примитив подтверждения **Conclude.confirm**, указывающий значение **Result(+)** MMS-пользователю. Если результирующий параметр показывает, что попытка вывода была неудачной, то механизм MMPM выдает примитив запроса **M-U-abort.request** и доставляет примитив подтверждения **Conclude.confirm**, указывающий значение **Result(+)** MMS-пользователю.

24.4 Непосредственно-отображеннная услуга прерывания

Услуга прерывания MMS непосредственно отображается на услугу **M-U-Abort.** При этом настоящий стандарт не дает определения прерывания PDU.

После получения примитива отображения (**M-U-abort** или **M-P-abort**) из поддерживающей системы связи, дающего спецификацию прерывания, механизм MMPM выдает примитив отображения прерывания MMS-пользователю. Если запрос прерывания генерируется системой, в которой находится MMS-пользователь (то есть прерывание **M-P-abort**), то сгенерированный по месту параметр примитива отображения прерывания MMS имеет значение **true**. В противном случае данный параметр имеет значение **false.**

После получения примитива запроса прерывания **Abort.request** от MMS-пользователя, механизм MMPM выдает примитив запроса **M-U-abort.request**.

Механизм MMPM может в любое время выдать примитив отображения прерывания **Abort. indication** MMS-пользователю и примитив запроса **M-P-abort.request** в качестве дополнения (в зависимости от обстоятельств).

24.5 Конструкция MMS PDU (блок данных протокола спецификации производственного сообщения)

После получения примитива запроса (ответа) для любой MMS-услуги, отличной от примитива запроса прерывания **Abort.request**, от примитива запроса заключения **Conclude.request** и от примитива ответа заключения **Conclude.response**, механизм MMPM должен:

a) создать блок данных протокола PDU в соответствии с разделом 7 для услуги, описанной в указанном примитиве в соответствии с требованиями протокола указанной услуги (см. разделы 7–23 и приложений C и D);

b) отправить созданный блок данных протокола PDU (как данные пользователя) на примитив услуг **M-DATA**, описанный выше, в соответствии с ИСО 9506-1 и настоящим стандартом.

24.6 Доставка примитивов услуг MMS-пользователю

После получения примитива отображения (подтверждения) от нижележащей системы связи, отличного от примитива **Abort.indication**, от примитива **M-RELEASE.indication** и примитива **M-RELEASE. confirm**, механизм MMPM определяет, содержит ли полученный примитив услуг корректный блок данных MMS PDU как данные пользователя. Блок данных MMS PDU является корректным, если он удовлетворяет требованиям абстрактного синтаксиса MMS при определении PDU, отображен на корректный примитив услуг (см. выше) и получен в соответствии со всеми правилами упорядочивания, определенными ИСО 9506-1 и настоящим стандартом.

Если полученный примитив услуг содержит корректный блок данных MMS PDU, то механизм MMPM выдает соответствующий примитив отображения (подтверждения) услуги со значениями примитива, соответствующими ИСО 9506-1 и настоящему стандарту.

Если полученный примитив услуги не содержит корректного блока данных MMS PDU, то механизм MMPM предпринимает следующие действия:

a) если блок данных запроса **Initiate-RequestPDU** и блок данных ответа **Initiate-ResponsePDU** успешно обмениваются посредством предшествующих связей в прикладной ассоциации, то механизм MMPM выдает отображение **reject.indication** MMS-пользователю, формирует блок данных отказа **RejectPDU** (с параметрами, основанными на выявленных ошибках) и отправляет данный PDU примитиву запроса **M-DATA.request**;

b) в противном случае механизм MMPM выдает отображение прерывания **Abort.indication** MMS-пользователю, а также выдает примитив запроса **M-abort.request**, если прикладная ассоциация доступна.

24.7 Право на отправку данных

Настоящий стандарт требует, чтобы нижележащие системы связи обеспечивали полную поддержку дуплексной связи.

24.8 Надежные услуги нижнего уровня

Настоящий стандарт не содержит указаний по обработке неупорядоченных сообщений, ошибок передачи, утерянных сообщений, а также дублируемых сообщений. В настоящем стандарте принято допущение о существовании надежных услуг нижнего уровня для данных обмена между двумя сущностями приложения.

24.9 Управление потоком (данных)

MMS не обеспечивает одноранговое управление потоком (данных). Получающий механизм MMPM может использовать имеющийся механизм управления потоком нижележащей системы, для того чтобы оказывать влияние на управление потоком (данных) всей прикладной ассоциации. Таким образом ограничиваются возможности одноранговых узла связи в части отправки данных. Решение о том, когда и как используют указанные возможности управления потоком, имеет локальный характер.

24.10 Использование контекстов представления данных

Взаимосвязь открытых систем OSI определяет контекст представления данных как абстрактный синтаксис, используемый приложением для связи с соответствующими одноранговыми узлами, и как синтаксис передачи, обеспечивающий механизм кодирования передаваемой информации. ИСО 9506 определяет только абстрактный синтаксис сообщений обмена между одноранговыми MMS-пользователями. Вопрос выбора необходимого синтаксиса передачи в настоящем стандарте не рассматривается.

Приложение А содержит рекомендации по использованию синтаксиса передачи в среде OSI. В других средах может быть использован синтаксис передачи, применяемый моделью OSI, а также другие соответствующие схемы кодирования. Единственным требованием к указанным синтаксисам передачи является наличие возможности переформирования абстрактного синтаксиса, используемого при однозначной спецификации производственного сообщения MMS с учетом имеющейся схемы кодирования.

Соглашение об использовании синтаксисов передачи для указанной реализации системы связи может оказаться важнейшим соглашением между заинтересованными сторонами. Оно также может быть предметом переговоров в части услуг нижнего уровня рассматриваемой системы связи.

24.11 Определение абстрактного синтаксиса

В настоящем стандарте идентификатору объекта ASN.1 присвоено значение:

{ iso standard 9506 part(2) mms-abstract-syntax-version1(1)}

как абстрактному синтаксису множества значений представления данных, каждое из которых является значением модуля ASN.1, определенного в настоящем стандарте (см. 7–23, приложения C, D и E) и в ИСО 9506-1 (см. 7–23). Соответствующее значение описателя объекта ASN.1 — это:

«mms-abstract-syntax-major-version1”

Основным номером данной версии ИСО 9506-1 и настоящего стандарта является № 1. Вспомогательным номером данной версии ИСО 9506-1 и настоящего стандарта является № 4.

В настоящем стандарте идентификатору объекта ASN.1 присвоено значение:

{ iso standard 9506 part(2) mms-file-record-version1(2)}

как абстрактному синтаксису множества значений представления данных, каждое из которых является значением модуля ASN.1, определенного в приложении B. Соответствующее значение описателя объекта ASN.1 равно «mms-file-record-version1».

25 Утверждение и конфигурация инициализации

25.1 Введение

Настоящий раздел описывает утверждения конфигурации и инициализации (CIS) для возможной практической реализации. Это позволяет разработчику фиксировать значения полей операторов на практике. Кроме того, это дает возможность инициализировать указанные поля при запуске системы. Каждый разработчик должен правильно составлять рассматриваемое утверждение CIS. В настоящем разделе рассмотрены отношения разработчика (поставщика оборудования) и программного обеспечения, необходимого для создания виртуальных устройств VMD, а также отношения разработчика и наладчика (собственника), отвечающего за наделение оборудования требуемыми свойствами.

25.2 CIS. Часть 1: инициализация виртуального производственного устройства VMD

Необходим доступ к информации по всем пунктам таблицы 1. Если рассматриваемый пункт требует дополнительной информации и пояснений, то разработчик делает ссылку в таблице на соответствующую страницу, раздел или параграф.

Рассматриваемая реализация должна поддерживать один объект из класса объектов виртуальных производственных приспособлений VMD. Каждому полю модели VMD разработчик ставит в соответствие начальное значение. Каждому предварительно определенному подчиненному объекту VMD разработчик ставит в соответствие значение в поле имени, а также либо (1) ссылается на полное определение объекта (например, на ИСО 9506-1 для объектов MMS), либо (2) дает его полное определение путем задания значений всех полей рассматриваемого объекта.

Таблица 1 — Информация по практической реализации CIS

Серийный № практической реализации: Дата:

Поле модели VMD	СВВ	Значение
&executiveFunction		
&vendorName		
&modelName		
&revision		
&AbstractSyntaxes		
&accessControl		
&Capabilities		См. таблицу 2
&local-detail		
&AccessControlList		См. таблицу 3
&Domains		См. таблицу 4
&ProgramInvocation		См. таблицу 5
&UnitControls		См. таблицу 6
&UnnamedVariables	vadr	См. таблицу 7
&NamedVariables	vnam	См. таблицу 8
&NamedVariableList	vlis	См. таблицу 9
&NamedTypes	vnam	См. таблицу 10
&DataExchanges		См. таблицу 11
&Semaphores		См. таблицу 12
&OperatorStations		См. таблицу 13
&EventCondition		См. таблицу 14
&EventAction		См. таблицу 15
&EventEnrollment		См. таблицу 16
&EventConditionList	cspi	См. таблицу 17
&Journal		См. таблицу 18
&selected-Program-Invocation	csr	

25.2.1 &executiveFunction (исполняемая функция)

Разработчик должен давать значение ссылки на приложение, идентифицирующее данную практическую реализацию в сетевой среде. Характер рассматриваемой ссылки (на приложение) зависит от сети, в которой находится разработчик. В сети OSI в данном поле указан идентификатор объекта.

25.2.2 &vendorName (имя поставщика)

Разработчик указывает значение символьной строки, идентифицирующей поставщика системы.

25.2.3 &modelName (имя модели)

Здесь разработчик указывает значение символьной строки, идентифицирующее модель рассматриваемой реализации.

25.2.4 &revision (пересмотр)

Разработчик указывает значение символьной строки, указывающее версию данной практической реализации.

25.2.5 &AbstractSyntaxes (абстрактные синтаксисы)

Разработчик указывает множество абстрактных синтаксисов, распознаваемых данной практической реализацией либо в операциях подкачки/загрузки области, либо в аргументах операций **Start** и **Resume,** либо в обоих случаях. Данное множество может быть пустым.

25.2.6 &EATransactions

Данному полю разработчик ставит в соответствие пустое множество.

25.2.7 &accessControl (управление доступом)

Разработчик указывает имя предварительно определенного объекта перечня средств управления доступом, описывающего характеристики управления доступом данного виртуального приспособления VMD. Настоящий объект рассмотрен в 25.2.12.

25.2.8 &logicalStatus (логический статус)

При старте системы данное поле также инициализируется. Это отражает условие работы оборудования, обеспечивающего VMD. Значение поля выбирается из четырех вариантов: **state-changes-allowed** (изменение состояние разрешено), **no-state-changes-allowed** (изменение состояния не разрешено), **limited-services-permitted** (использование услуг ограничено), **support-services-allowed** (разрешено использование услуг поддержки)**.**

25.2.9 &Capabilities (возможности)

Разработчик заполняет таблицу 2, дающую описание каждой возможности виртуального приспособления VMD. Нумерация строк таблицы расширяется при необходимости расширенного описания указанных возможностей.

Данная таблица может быть пустой.

Таблица 2 — Описание возможностей

Возможности (символьная строка)	Смысл	Грамматическое правило

25.2.10 &physicalStatus (физический статус)

При старте системы данное поле также инициализируется. Это отражает условия работы оборудования, обеспечивающего функционирование VMD. Значение поля выбирается из четырех вариантов: **operational** (в рабочем состоянии), **partially-operational** (возможности ограничены), **inoperable** (в нерабочем состоянии), **needs-commissioning** (требует списания)**.**

25.2.11 &local-detail (местные особенности)

Разработчик указывает начальное значение битовой строки локальных особенностей, а также смысл каждого бита. Если биты локальных особенностей не определены, то выбирается значение **B**.

25.2.12 &AccessControlList (перечень средств управления доступом)

Разработчик указывает множество предварительно определенных объектов перечня средств управления доступом. Для каждого указанного объекта он заполняет таблицу 3. При этом либо дана ссылка на соответствующее определение, либо указаны значения для всех элементов поля объекта.

Данное множество содержит по крайней мере один объект. На него дана ссылка в поле **&accessControl** таблицы 1.

Таблица 3 — Предварительно определенный объект управления доступом

Объект перечня средств управления доступом	СВВ	Значение или ссылка
&name		
Reference to Definition		
&accessControl		
&readAccessCondition		
&storeAccessCondition		
&writeAccessCondition		
&loadAccessCondition		
&executeAccessCondition		
&deleteAccessCondition		
&editAccessCondition		
&AccessControlList		
&Domains		
&ProgramInvocation		
&UnitControls		
&UnnamedVariables	vadr	
&NamedVariables	vnam	
&NamedVariableList	vlis	
&NamedType	vnam	
&DataExchanges		
&Semaphores		
&OperatorStations		
&EventCondition		
&EventAction		
&EventEnrollment		
&Journal		
&EventConditionList	cspi	

25.2.13 &Domains (области)

Разработчик указывает множество предварительно определенных объектов области. Для каждого указанного объекта он заполняет таблицу 4. При этом либо дана ссылка на определение, либо указаны значения для всех элементов полей объектов.

Для поля **&Capabilities** формируется одна или несколько копий таблицы 2. Эти копии регистрируют возможность, ассоциированную с указанной предварительно определенной областью. Поле **&state** имеет значения **ready** или **in-use** в зависимости от того, является ли поле вызова программы **&ProgramInvocation** пустым или нет. Поле **&aAssociation** должно быть пустым.

Для каждого предварительно определенного объекта, подчиненного указанной области, таблица, соответствующая рассматриваемому типу объекта, формируется в поле **&content** таблицы 4, и на нее производится ссылка.

Данное множество может быть пустым.

Таблица 4 — Предварительно определенный объект области

Объект области	СВВ	Значение или ссылка
&name		
Reference to Definition		
&Capabilities		
&state		
&aAssociation		empty (пусто)
&accessControl		
&sharable		
&ProgramInvocation		
&uploadsInProgress		Равно 0

Окончание таблицы 4

Объект области	CBB	Значение или ссылка
&NamedVariables	vnam	См. таблицу 8
&NamedVariableList	vlis	См. таблицу 9
&NamedType	vnam	См. таблицу 10
&EventCondition		См. таблицу 14
&EventAction		См. таблицу 15
&EventEnrollment		См. таблицу 16
&EventConditionList	cspi	См. таблицу 17

25.2.14 &ProgramInvocation (активизация программы)

Разработчик указывает множество предварительно определенных объектов вызова программы. Для каждого объекта он заполняет таблицу 5. При этом либо дана ссылка на определение, либо указаны значения для всех элементов полей объекта.

Данное множество может быть пустым. Если данное множество не пусто, то таблица 4 заполняется.

Таблица 5 — Предварительно определенный объект вызова программы

Объект вызова программы	CBB	Значение или ссылка
&name		
reference to Definition		
&ProgramInvocationState		
&Domains		
&accessControl		
&reusable		
&monitor		
&EventCondition		
&EventAction		
&EventEnrollment		
&executionArgument		
&control	csr	
&controlling-Program-Invocation	csr	
&controlled-Program-Invocation	csr	

25.2.15 &UnitControls (блоки управления)

Разработчик указывает множество предварительно определенных объектов блока управления. Для каждого объекта он заполняет таблицу 6. При этом либо дана ссылка на определение, либо указаны значения для всех элементов полей рассматриваемого объекта.

Данное множество может быть пустым. Если данное множество не пусто, то таблица 4 также заполняется.

Таблица 6 — Предварительно определенный объект блока управления

Объект блока управления	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&Domains		
&ProgramInvocation		

25.2.16 &UnnamedVariables (непоименованные переменные)

Если практическая реализация может поддерживать структурный элемент согласованности **vadr**, то разработчик заполняет таблицу 7. При этом предоставляется информация о составе непоименованных переменных, включая их формат, диапазон поддерживаемых адресов [например, указывается их тип: **numeric** (число), **symbolic** (символ), **unconstrained** (произвольное)] и алгоритмы ассоциации выбора типа описания с адресом.

Таблица 7 — Объекты непоименованных переменных

Непоименованные переменные	Описание
Address	
Type description	

25.2.17 &NamedVariables (поименованные переменные)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **vnam**, то разработчику следует указать множество предварительно определенных объектов поименованных переменных. Для каждого объекта разработчик заполняет таблицу 8, в которой он либо дает ссылку на определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Данное множество может быть пустым.

Таблица 8 — Предварительно определеный объект поименованной переменной

Объект поименованной переменной	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&typeDescription		
&accessMethod		
&address	vadr	
&meaning	sem	

25.2.18 &NamedVariableList (перечень поименованных переменных)

Если рассматриваемая практическая реализация поддерживает структурные элементы согласованности **vnam** и **vlis**, то разработчику следует указать множество предварительно определенных объектов перечня поименованных переменных. Для каждого объекта разработчик заполняет таблицу 9, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

В каждом пункте поля **&listOfVariables** разработчик идентифицирует ссылочные поименованные или непоименованные переменные, а также дает спецификацию альтернативного доступа **AlternateAccess** (при его наличии).

Данное множество может быть пустым.

Таблица 9 — Предварительно определенный объект перечня поименованных переменных

Объект перечня поименованных переменных	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&listOfVariable		
unnamedItem	vadr	
namedItem	vnam	
alternateAccess	valt	

25.2.19 &NamedType (поименованный тип)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **vnam**, то разработчику следует указать множество предварительно определенных объектов поименованного типа. Для каждого объекта разработчик заполняет таблицу 10, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Данное множество может быть пустым.

Таблица 10 — Предварительно определенный объект поименованного типа

Объект поименованного типа	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&typeDescription		
&meaning	sem	

25.2.20 &DataExchanges (обмен данными)

Разработчик указывает множество предварительно определенных объектов обмена данными. Для каждого объекта он заполняет таблицу 11, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Для каждого объекта поле **&inUse** инициализируется со значением **false.**

Данное множество может быть пустым.

Таблица 11 — Предварительно определенный объект обмена данными

Объект обмена данными	СВВ	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&request		
&response		
&linked		
&ProgramInvocation		

25.2.21 &Semaphore (Семафор)

Разработчик указывает множество предварительно определенных объектов типа «семафор». Для каждого объекта он заполняет таблицу 12, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта. В зависимости от значения параметра **&class**, значения параметров **&numberOfTokens** или **&Named-Tokens** могут не указывать.

Поля **&numberOfOwnedTokens** (при условии их наличия) имеют начальные нулевые значения. Поля **&Owners** и **&Requesters** инициализируются как пустое множество.

Данное множество может быть пустым. Если данное множество не пусто, то поле **&EventCondition** также должно быть заполнено.

Таблица 12 — Предварительно определенный объект типа «семафор»

Объект типа «семафор»	СВВ	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&class		
&numberOfTokens		
&Named-Tokens		
&EventCondition		

25.2.22 &OperatorStations (станция управления)

Разработчик указывает множество предварительно определенных объектов станции управления. Для каждого такого объекта он заполняет таблицу 13, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Поле **&inputBuffer** (при условии его наличия) инициализируется как пустая строка, так же как и поле **&OutputBuffers**. Поле **&state** инициализируется как **idle.**

Данное множество может быть пустым.

Таблица 13 — Предварительно определенный объект станции управления

Объект станции управления	СВВ	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&stationType		

25.2.23 &EventCondition (условие события)

Разработчик указывает множество предварительно определенных объектов условия события. Для каждого объекта он заполняет таблицу 14, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Поле **&timeToActive** и поле **&timeToIdle** инициализируются как **undefined.**

Данное множество может быть пустым.

Таблица 14 — Предварительно определенный объект условия события

Объект условия события	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&ecClass		
&ecState		
&Priority		
&severity		
&EventEnrollment		
&enabled		
&AlarmSummaryReports		
&monitoredVariable		
&evaluationInterval		
&displayEnhancement	cspi	
&group-Priority-Override	cspi	
&ReferencingEventConditionList	cspi	

25.2.24 &EventAction (действие события)

Разработчик указывает множество предварительно определенных объектов действия события. Для каждого объекта он заполняет таблицу 15, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Поле **&timeToActive** и поле **&timeToIdle** инициализируются как **undefined.**

Данное множество может быть пустым.

Таблица 15 — Предварительно определенный объект действия события

Объект действия события	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&ConfirmedServiceRequest		
&Modifiers		
&EventEnrollment		

25.2.25 &EventEnrollment (регистрация события)

Здесь разработчик указывает множество предварительно определенных объектов регистрации события. Для каждого объекта разработчик заполняет таблицу 16, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Поле **&aAssociation** инициализируется как пустое. Поле **&InvokeID** (при условии его наличия) инициализируется как нуль. Поле **&notificationLost** (при его наличии) инициализируется со значением **false**. Поле **&timeActiveAck** и поле **&timeIdleAck** (при условии их наличия) инициализируются как **undefined**. Поле **&ackState** (при его наличии) инициализируется как **acked**.

Данное множество может быть пустым. Если данное множество не пусто, то множество **&EventCondition** также не должно быть пустым.

Таблица 16 — Предварительно определенный объект регистрации события

Объект регистрации события	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&eeClass		
&EventCondition		
&ecTransitions		

Окончание таблицы 16

&remainingDelay		
&EventAction		
&duration		
&ClientApplication		
&aaRule		
&displayEnhancement	cspi	

25.2.26 &EventConditionList (перечень условий события)

Здесь разработчик указывает множество предварительно определенных объектов перечня условий события. Для каждого объекта разработчик заполняет таблицу 17, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Данное множество может быть пустым. Если данное множество не пусто, то множество **&EventCondition** также не должно быть пустым.

Таблица 17 — Предварительно определенный объект перечня условий события

Объект перечня условий события	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&EventCondition		
&EventConditionList	recl	
&ReferencingEventConditionList	recl	

25.2.27 &Journal (журнал)

Разработчик указывает множество предварительно определенных объектов журнала. Для каждого объекта разработчик заполняет таблицу 18, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Данное множество может быть пустым.

Таблица 18 — Предварительно определенный объект журнала

Объект журнала	CBB	Значение или ссылка
&Name		
reference to Definition		
&accessControl		
&Entries		

Для каждой записи в поле **&Entries** рассматриваемого объекта журнала разработчик заполняет таблицу 19, в которой либо дает ссылку на соответствующее определение, либо указывает значения для всех элементов полей рассматриваемого объекта.

Значения, указываемые в поле **&entry,** должны быть уникальными внутри данного журнала.

Данное множество может быть пустым.

Таблица 19 — Предварительно определенный объект журнальной записи

Объект журнальной записи	CBB	Значение или ссылка
&Journal		
&entry		
&ClientApplication		
&timestamp		
&orderOfReceipt		
&informationType		
&textComment		
&eventTransitionsRecord		
&journalVariables		

25.2.28 &operation-State (рабочее состояние)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **csr**, то разработчик инициализирует данное поле, для того чтобы указать состояние оборудования, обеспечивающего работу VMD. Возможные значения инициализации: **idle** (ожидание), **loaded** (загружен), **ready** (готов к работе), **execution** (выполнение), **motion-paused** (пауза), а также **manualInterventionRequired** (необходимо вмешательство оператора).

25.2.29 &safety-Interlocks-Violated (замки безопасности взломаны)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **csr**, то разработчик инициализирует соответствующее поле, для того чтобы указать состояние оборудования, обеспечивающего работу VMD.

25.2.30 &any-Resource-Power-On (подключение к произвольному источнику тока)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **csr**, то разработчик инициализирует соответствующее поле, для того чтобы указать состояние оборудования, обеспечивающего работу VMD.

25.2.31 &local-Control (локальное управление)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **csr**, то разработчик инициализирует соответствующее поле, для того чтобы указать состояние оборудования, обеспечивающего работу VMD.

25.2.32 &selected-Program-Invocation (активизация выбранной программы)

Если рассматриваемая практическая реализация поддерживает структурный элемент согласованности **csr**, то разработчику следует указать начальное значение параметра активизации выбранной программы **selected-Program-Invocation**. Данное значение может быть нулем.

25.3 CIS. Часть 2: CBB (структурные элементы согласованности) услуг и параметров

В следующих таблицах разработчик предоставляет информацию о структурных элементах согласованности (CBB) услуг и параметров, поддерживаемых рассматриваемой практической реализацией. Данная информация указывает, выполняет ли рассматриваемая реализация требования сервера, требования клиента или оба требования одновременно при использовании абстрактного синтаксиса, определенного в настоящем стандарте. Термины «клиент», «сервер», «запрос» и «ответ» определены в разделе 5 ИСО 9506-1. Требования сервера и требования клиента каждого структурного элемента согласованности (CBB) описаны в разделе 25 ИСО 9506-1.

25.3.1 Среда и общие услуги управления

В таблице 20 (только для данного пункта) указано наличие поддержки. Необходимо указать, может ли данная практическая реализация поддерживать роль запрашивающего устройства, роль ответчика или обе роли одновременно.

Таблица 20 — Среда и общие услуги управления

Среда и общее управление	Запрашивающее устройство	Ответчик
Инициировать		
Завершить		
Отменить		

В таблице 21 разработчик указывает:

1) наличие поддержки CBB следующих параметров: **char, csr, csnc, csplc, cspi**;

2) вызываемые/вызванные локальные детали. Разработчик указывает деталь, использованную в параметрах **LocalDetailCalling** и **LocalDetailCalled**, применяемую в услуге запуска **Initiate**, если эти параметры являются частью рассматриваемой практической реализации. Семантика указываемых значений должна соответствовать установленным требованиям;

3) уровень поддержки времени. Разработчик указывает, что необходимо для функционирования часов системы: дата или время дня, а также необходимость поддержки идентификатора временной последовательности;

4) необходимость дробления времени до миллисекунд. Разработчик указывает наименьшую необходимую единицу времени (выраженную в миллисекундах), определяющую разрешение процессов во времени. Данное значение имеет смысл, только если поддерживаются дата и время дня.

Таблица 21 — Среда и общие параметры управления

Общие параметры управления	Значение
Char	
Csr	
Csnc	
Csplc	
Cspi	
Local Detail (местные детали)	
Support for time (поддержка времени)	
Granularity of time (ms) (степень дробления времени)	

25.3.2 Услуги управления доступом

В таблице 22 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 22 — Услуги управления доступом

Управление доступом	Сервер	Клиент
Определение перечня средств управления доступом		
Получение атрибутов перечня средств управления доступом		
Регистрация объектов с управляемым доступом		
Удаление перечня средств управления доступом		
Изменение управления доступом		

В таблице 23 разработчик указывает наличие поддержки **aco CBB**.

Таблица 23 — Параметр управления доступом

Параметр управления доступом	
Aco	

25.3.3 Услуги поддержки VMD

В таблице 24 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 24 — Услуги поддержки VMD

Поддержка VMD	Сервер	Клиент
Статус		
Незапрашиваемый статус		
Получение перечня имен		
Идентифицировать **Identify**		
Переименовать		
Получение перечня возможностей		
Останов VMD		
Перезагрузка VMD		

Практическая реализация, отображающая поддержку ответчика для услуги запуска **Initiate,** необходима для поддержки роли сервера для услуги идентификации **Identify**.

В таблице 25 разработчик указывает:

1) параметр **Local Detail** (параметр услуг **Status** и **UnsolicitedStatus**): разработчик указывает, что представляет собой рассматриваемая локальная подробность и как данный параметр грамматически оформлен внутри битовой строки (синтаксис и семантика символьной строки должны удовлетворять установленным требованиям);

2) метод расширенного вывода статуса информации: разработчик указывает метод расширенного вывода статуса информации, при его наличии (см. 10.3).

Таблица 25 — Параметр поддержки виртуального приспособления VMD

Параметр поддержки VMD	Значение
Local Detail	
Extended Derivation	

25.3.4 Услуга управления доменом

В таблице 26 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 26 — Услуги управления доменом

Поддержка управления доменом	Сервер	Клиент
Инициирование последовательности загрузки		
Сегмент загрузки		
Завершение последовательности загрузки		
Инициирование последовательности подкачки		
Сегмент подкачки		
Завершение последовательности загрузки		
Запрос загрузки области		
Запрос подкачки области		
Загрузка контента области		
Хранение контента области		
Удаление области		
Получение атрибутов области		

В таблице 27 разработчик указывает:

1) поддерживает ли рассматриваемая практическая реализация **CBB** с параметром **tpy**;

2) формат загрузки данных. Разработчик дает семантические и синтаксические определения октетной строки в параметре загрузки данных услуг **DownloadSegment** и **UploadSegment**;

3) если поддерживается выбор **EXTERNAL** или **EMBEDDED PDV** для указанных параметров, то разработчик указывает поддерживаемые имена абстрактного синтаксиса;

4) максимальное количество механизмов подкачки. Здесь разработчик указывает максимальное количество механизмов подкачки, одновременно задействуемых в одной области.

Таблица 27 — Параметры управления доменом

Параметры области	Значение
tpy	
Загрузка данных – октетные строки	
Загрузка данных – абстрактный синтаксис	
Максимальное количество механизмов подкачки	

25.3.5 Услуги управления активизацией программы

В таблице 28 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 28 — Услуги управления активизацией программы

Поддержка управления активизацией программы	Сервер	Клиент
Формирование процесса вызова программы		
Удаление вызова программы		
Начать		
Останов		
Возобновить выполнение		
Перезагрузка		
Аннулирование		
Получение атрибутов вызова программы		
Выбор		
Изменение атрибутов вызова программы		
Реконфигурация вызова программы		

В таблице 29 разработчик указывает:

1) аргумент выполнения (параметр услуг **start** и **Reset**): определено максимальное количество поддерживаемых символов символьной строки параметра аргумента выполнения;

2) грамматические правила для символьной строки аргументов выполнения;

3) если для данного параметра поддерживается выбор **EXTERNAL** или **EMBEDDED PDV**, то приведены поддерживаемые имена абстрактного синтаксиса;

4) формат обозначения для поля **&programLocation** (при его наличии);

5) поддерживается ли шаговый режим работы **StepMode.**

Таблица 29 — Параметры управления активизацией программы

Параметры вызова программы	Значение
Максимальный размер аргумента выполнения	
Грамматические правила для аргументов выполнения	
Абстрактный синтаксис аргумента выполнения	
&programLocation	
StepMode	

25.3.6 Услуги блока управления

В таблице 30 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 30 — Услуги блока управления

Поддержка управления блока управления	Сервер	Клиент
Инициирование загрузки блока управления		
Сегмент загрузки блока управления		
Подкачка блока управления		
Пуск блока управления		
Останов блока управления		
Создание блока управления		
Добавление к блоку управления		
Удаление из блока управления		
Получение атрибутов блока управления		
Загрузка блока управления из файла		
Хранение блока управления в файле		
Удаление блока управления		

25.3.7 Услуги доступа к переменной

В таблице 31 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 31 — Услуги доступа к переменной

Поддержка доступа к переменной	Сервер	Клиент
Читать		
Писать		
Информационный отчет		
Получение атрибутов доступа к переменной		
Определение поименованной переменной		
Удаление доступа к переменной		
Определение перечня поименованных переменных		
Получение атрибутов перечня поименованных переменных		
Удаление перечня поименованных переменных		
Определение поименованного типа		
Получение атрибутов поименованного типа		
Удаление поименованного типа		

В таблице 32 разработчик указывает:

1) значения **CBB** параметров **str1, str2, vnam, vadr, valt, vlis**;

2) максимальное значение поддерживаемого параметра **nest**;

3) если рассматриваемая практическая реализация поддерживает механизм, описанный в ИСО 9506-1, то следует указать **CBB** параметров **real**;

4) порядок обеспечения непрерываемого доступа к переменной: определено, при каких обстоятельствах можно гарантировать непрерываемый доступ к переменной, а также приведен порядок достижения данного уровня с помощью MMS-услуг;

5) если поддерживается режим **vadr**, то разработчик указывает, поддерживается ли режим **SINGLE**, режим **UNNAMED** или оба эти режима спецификации переменной (см. 12.5.2.1 ИСО 9506-1).

Таблица 32 — Параметры доступа к переменной

Параметры доступа к переменной	Значение
str1	
str2	
vnam	
vadr	
valt	
vlis	
nest	
Uninterruptible access	
SINGLE	
UNNAMED	

В таблице 33 разработчик указывает возможные значения поля **Size** типа данных, поддерживаемых как сервер.

Таблица 33 — Параметры данных

Данные — параметр Size	Значение
Битовая строка	
Целое	
без знака	
С плавающей точкой	
Октетная строка	
Видимая строка	
Бинарное время	
bcd	
mMSString	

25.3.8 Услуги обмена данными

В таблице 34 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 34 — Услуги обмена данными

Поддержка обмена данных	Сервер	Клиент
Обмен данных		
Получение атрибута обмена данными		

25.3.9 Услуги управления семафором

В таблице 35 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 35 — Услуги управления семафором

Поддержка семафора	Сервер	Клиент
Получение управления		
Освобождение управления		
DefineSemaphore (определить семафор)		

Окончание таблицы 35

Поддержка семафора	Сервер	Клиент
DeleteSemaphore (стереть семафор)		
Отчет о статусе семафора		
Отчет о статусе группового семафора		
Отчет о статусе записей семафора		
Прикрепление к модификатору семафора		

В таблице 36 разработчик указывает приоритетную операцию **Priority processing** семафора. Разработчик указывает алгоритм определения приоритета для семафора (если семафоры это поддерживают). Параметр степени дробления времени представлен в таблице 25.

Таблица 36 — Параметры управления семафором

Операция **Priority processing**	Описание
Алгоритм	

25.3.10 Услуги связи оператора

В таблице 37 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли одновременно для следующих услуг.

Таблица 37 — Услуги связи оператора

Поддержка станции управления	Сервер	Клиент
Вход		
Выход		

В таблице 38 разработчик указывает максимальное значение параметра **Input time out** (истечение указанного времени, с).

Таблица 38 — Параметр связи оператора

Параметр станции управления	Значение
Input time out	

25.3.11 Услуги управления событием

В таблице 39 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 39 — Услуги управления событием

Поддержка управления событием	Сервер	Клиент
Trigger Event (запуск события)		
EventNotification (уведомление о событии)		
Подтверждение **EventNotification**		
Получение сводки о сигнале тревоги **AlarmSummary**		
Получение сводки о регистрации сигнала тревоги **AlarmEnrollmentSummary**		
Прикрепление к модификатору условия события		

25.3.12 Услуги условия события

В таблице 40 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 40 — Услуги условия события

Поддержка управления условия события	Сервер	Клиент
Определение условия события		
Удаление условия события		
Получение атрибутов условия события		
Отчет о статусе условия события		
Мониторинг измений условия события		

В таблице 41 разработчик указывает, поддерживает ли рассматриваемая практическая реализация **СВВ** параметры **cei, des, dei**.

Таблица 41 — Параметры условия события

Параметры условия события	Значение
cei	
des	
dei	

Примечание — СВВ параметры **des** и **dei** также используются для услуг регистрации события.

25.3.13 Услуги действия события

В таблице 42 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 42 — Услуги действия события

Поддержка управления действием события	Сервер	Клиент
Определение действия события **EventAction**		
Удаление действия события		
Получение атрибутов действия события		
Отчет о статусе условия события		
Отчет о статусе действия события		

25.3.14 Услуги регистрации события

В таблице 43 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 43 — Услуги регистрации события

Поддержка управления регистрацией события	Сервер	Клиент
Определение регистрации события		
Удаление регистрации события		
Получение атрибутов регистрации события		
Отчет о статусе регистрации события		
Изменение регистрации события		

25.3.15 Услуги перечня условий события

В таблице 44 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 44 — Услуги перечня условий события

Поддержка перечня условий события	Сервер	Клиент
Определение перечня условий события		
Удаление перечня условий события		
Добавление ссылки перечня условий события		
Удаление ссылки перечня условий события		
Получение атрибутов перечня условий события		
Отчет о статусе перечня условий события		
Мониторинг изменений перечня условий события		

В таблице 45 разработчик указывает, поддерживает ли рассматриваемая практическая реализация **СВВ** параметр **recl**.

Таблица 45 — Параметр перечня условий события

Параметр перечня условий события	Значение
Recl	

25.3.16 Услуги управления журналом

В таблице 46 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 46 — Услуги управления журналом

Поддержка управления журналом	Сервер	Клиент
Читать журнал		
Писать журнал		
Инициализировать журнал		
Отчет о статусе журнала		
Создание журнала		
Удаление журнала		

25.3.17 Ошибки

В таблице 47 разработчик указывает:

1) дополнительный код (параметр типа ошибки): приведены коды, используемые параметром дополнительного кода. Следует указать целые коды и их смысл;

2) дополнительные подробности (параметр типа ошибки): указаны подробности, используемые параметром дополнительных подробностей. Следует определить минимальное и максимальное число октетов, используемых для символьных строк, а также синтаксис и семантики данных символьных строк.

Таблица 47 — Параметры ошибки

Параметры ошибки	Значение	смысл
Дополнительный код		
Дополнительные подробности		

25.3.18 Услуги доступа к файлу

В таблице 48 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 48 — Услуги доступа к файлу

Поддержка доступа к файлу	Сервер	Клиент
Получение файла		

25.3.19 Услуги управления файлом

В таблице 49 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 49 — Услуги управления файлом

Поддержка управления файлом	Сервер	Клиент
Открыть файл		
Читать файл		
Закрыть файл		
Переименовать файл		
Стереть файл		
Директория файла		

В таблице 50 разработчик указывает:

1) синтаксис имени файла: приведен синтаксис имен файлов, используемый в локальном файлохранилище. Следует указать разграничивающие обозначения для рассматриваемых иерархических файловых систем, а также приемлемые и неприемлемые наборы символов для имен файлов, ограничения на длину файла (при их наличии).

Таблица 50 — Параметры управления файлом

Параметры управления файлом	синтаксис	символы	Длина
Имя файла			

25.3.20 Услуги рассеянного доступа

в таблице 51 разработчик указывает, поддерживает ли рассматриваемая практическая реализация роль сервера, роль клиента или обе роли для следующих услуг.

Таблица 51 — Услуги рассеянного доступа

Поддержка рассеянного доступа	Сервер	Клиент
Определение рассеянного доступа		
Получение рассеянных атрибутов доступа		

Если рассматриваемая практическая реализация поддерживает любые возможности, описанные в приложении Е, то необходимо обеспечить поддержку СВВ параметра **vsca**.

В соответствии с разделом 25.3.20 разработчик указывает, поддерживает ли данная практическая реализация СВВ параметр **vsca**.

Таблица 52 — Параметры рассеянного доступа

Параметры рассеянного доступа	Значение
vsca	

Приложение А
(обязательное)

Связь *M*-услуг с сервисным элементом управления ассоциацией (ACSE) и услугами представления данных

В настоящем приложении описана возможность использования сервисного элемента управления ассоциацией (ACSE) и услуг уровня представления данных для реализации *M*-услуг, необходимых для работы механизма разработки протокола обмена сообщениями (MMPM). Любое использование услуг ACSE или услуг представления данных, отличное от описанного в настоящем приложении, приводит к ошибкам протокола.

Протокол сообщения MMS находится в среде взаимосвязи открытых систем и внутри прикладного уровня. Настоящий стандарт использует (отображает) услуги и примитивы услуг ACSE и уровня представления данных в качестве прикладного сервисного элемента (ASE). MMS-пользователи могут быть элементами рассматриваемого прикладного процесса или быть другими элементами услуг ASE.

А.1 Отображение *M*-услуги

Все блоки данных MMS PDU следует рассматривать как данные пользователя на элементе ACSE или на примитиве услуги представления данных. Отображение *M*-услуги на элементы ACSE и на услуги представления данных имеет следующий вид:

M-Service	ACSE or Presentation Service
M-ASSOCIATE	*A*-ASSOCIATE
M-RELEASE	*P*-DATA, A-RELEASE
M-DATA	*P*-DATA
M-U-ABORT	*A-U*-ABORT
M-P-ABORT	*A-P*-ABORT

А.1.1 Услуга *M*-ASSOCIATE

Примитивы услуг ***M*-ASSOCIATE** в точности соответствуют примитивам услуги ***A*-ASSOCIATE** (см. ИСО/МЭК 8649). Соответствие данных параметров рассмотрено ниже.

Параметры **Calling AP Title**, **Calling AE Qualifier**, **Calling AP Invocation-identifier** и **Calling AE Invocation-identifier** сущности **AARQ-apdu** указаны для выбора значения параметра **Calling Application Reference** услуги ***M*-ASSOCIATE**.

Параметры **Called AP Title**, **Called AE Qualifier**, **Called AP Invocation-identifier** и **Called AE Invocation-identifier** сущности **AARQ-apdu** указаны для выбора значения параметра **Called Application Reference** услуги ***M*-ASSOCIATE**.

Параметры **Responding AP Title**, **Responding AE Qualifier**, **Responding AP Invocation-identifier** и **Responding AE Invocation-identifier** сущности **AARE-apdu** указаны для выбора значения параметра **Responding Application Reference** услуги ***M*-ASSOCIATE**.

Следует использовать параметр **Authentication Value** сущности **AARQ-apdu** (при ее наличии) для того, чтобы указать значение поля **Authentication Value** аргумента услуги ***M*-ASSOCIATE**.

Следует указать значение параметра **Authentication Value** сущности **AARE-apdu** (при ее наличии) для того, чтобы указать значение поля **Authentication Value** результата услуги ***M*-ASSOCIATE**.

Следует использовать параметр **User Data** сущности **AARQ-apdu** для выбора значения поля **User Data** аргумента услуги ***M*-ASSOCIATE**.

Следует использовать параметр **User Data** сущности **AARE-apdu** для выбора значения поля **User Data** результата услуги ***M*-ASSOCIATE**.

Могут быть использованы другие параметры сущности **AARQ-apdu** для выбора значения поля **Other Communication Parameters** аргумента услуги ***M*-ASSOCIATE**.

Могут быть использованы другие параметры сущности **AARE-apdu** для выбора значения поля **Other Communication Parameters** результата услуги ***M*-ASSOCIATE**.

А.1.2 Услуга *M*-RELEASE

В среде **OSI** услуга ***M*-RELEASE** реализована как последовательность услуги ***P*-DATA** и услуги ***A*-RELEASE**. Процедура и соответствие параметров установленным требованиям рассмотрено ниже.

А.1.2.1 Запрашивающий MMS-пользователь

Если механизм MMPM получает примитив запроса **Conclude.request** от запрашивающего MMS-пользователя, то данный механизм MMPM выдает примитив запроса ***P*-DATA** с блоком данных **Conclude-RequestPDU** в качестве поля **UserField**,

После получения отображения ***P*-DATA**, содержащего корректный блок данных **Conclude-ResponsePDU**, механизм MMPM выдает примитив услуги **ACSE *A*-RELEASE.request**, не содержащий данных пользователя.

После получения примитива подтверждения ***A*-RELEASE.confirm** ACSE (данные пользователя которого игнорируются) с результирующим параметром, отображающим успешный выпуск прикладной ассоциации, механизм MMPM доставляет примитив подтверждения **Conclude.confirm**, указывающий значение **Result(+)** для MMS-пользователя. Если результирующий параметр указывает, что попытка вывода неудачна, то механизм MMPM выдает примитив запроса ***A*-abort.request** ACSE и доставляет примитив подтверждения **Conclude.confirm**, указывающий значение **Result(+)** для MMS-пользователя.

А.1.2.2 Ответающийся MMS-пользователь

После получения отображения ***P*-DATA**, содержащего корректный блок данных **Conclude-RequestPDU**, механизм MMPM выдает примитив отображения **Conclude.indication** для ответающегося MMS-пользователя.

После получения примитива ответа **Conclude.response**, указывающего факт приема сообщения ответающимся MMS-пользователем, механизм MMPM выдает примитив запроса ***P*-DATA**, содержащего блок данных **Conclude-ResponsePDU** в качестве поля **UserField**.

После получения примитива отображения ***A*-RELEASE.indication** ACSE (данные пользователя которого игнорируются), механизм MMPM выдает примитив ответа ***A*-RELEASE.response** ACSE, не содержащий данных пользователя. Данное сообщение содержит результирующий параметр, указывающий на успешный выпуск прикладной ассоциации. Механизм MMPM выдает примитив ответа **Conclude.response**, содержащий сообщение об успехе, запрашивающему MMS-пользователю.

После получения примитива ответа **Conclude.response**, указывающего на отказ ответающегося MMS-пользователя, механизм MMPM выдает примитив запроса ***P*-DATA**, содержащего блок данных **Conclude-ErrorPDU** в качестве поля **UserField**. Механизм MMPM выдает примитив ответа ***A*-RELEASE.response** ACSE, не содержащий данных пользователя. Это сообщение содержит результирующий параметр, указывающий на неудачный выпуск прикладной ассоциации. Механизм MMPM выдает также примитив ответа **Conclude.response**, указывающий на неудачу, запрашивающему MMS-пользователю.

А.2 Услуга *M*-DATA

Услуга ***M*-DATA** отображается прямо на услугу ***P*-DATA**. Параметр данных пользователя услуги ***M*-DATA** — это параметр данных пользователя услуги ***P*-DATA**.

А.3 Услуга *M-U*-Abort

Услуга прерывания ***M-U*-Abort** отображается прямо на услугу прерывания ***A-U*-Abort**. Параметр источника прерывания услуги ***M-U*-Abort** — это параметр источника прерывания услуги ***A-U*-Abort**.

А.4 Услуга *M-P*-Abort

Услуга ***M-P*-Abort** отображается прямо на услугу ***A-P*-Abort**. Параметр источника прерывания услуги ***M-P*-Abort** — это параметр источника прерывания услуги ***A-P*-Abort**.

А.5 Использование контекстов представления данных

Модель OSI поддерживает процедуру согласования контекста представления данных во время создания ассоциации. В отличие от первого издания настоящий стандарт определяет только один абстрактный синтаксис. Поэтому проблема множественности абстрактных синтаксисов в настоящем стандарте не актуальна. Однако настоящий стандарт может быть также использован в условиях множественности синтаксисов передачи, а работа системы связи в среде MMS и OSI требует наличия абстрактного синтаксиса ACSE. Поэтому согласование контекстов представления данных для заданной ассоциации оказывается необходимым условием.

Для рассмотрения одного или нескольких контекстов представления данных необходимо использовать параметр перечня определений контекста представления данных для примитива запроса ***A*-Associate.request** ACSE путем вызова сущности приложения **AE**. Аналогично параметр перечня результатов контекста представления данных для примитива ответа ***A*-Associate.response** ACSE может быть использован путем ответа сущности приложения **AE** на сообщение о приеме или отказе от предложенных элементов (см. ИСО/МЭК 8649 и ИСО 8822).

А.6 Определение синтаксиса передачи

Значение идентификатора объекта и значение описателя объекта ASN.1 имеют вид

{ joint-iso-ccitt asn1(1) basic-encoding(1) }

и

«Basic Encoding of a single ASN.1 type»

(они назначаются информационному объекту в соответствии со ИСО/МЭК 8825). Указанные значения могут быть использованы в качестве синтаксиса передачи вместе с рассматриваемым абстрактным синтаксисом.

А.7 Имя контекста приложения

Для использования приложений, содержащих объекты ACSE и MMS как элементы услуги приложения ASE, следующие значения идентификатора объекта

{ iso standard 9506 part(2) mms-application-context-version1(5) }

и значение описателя объекта:

«ISO MMS»

назначаются для информационного объекта типа:

«ACSE-1.ApplicationContextName»

в соответствии с ИСО/МЭК 8650. Несмотря на то что данный идентификатор объекта определен в настоящем стандарте и, следовательно, включает фрагмент «part(2)», данное имя контекста приложения должно ссылаться на требования ИСО 9506-1 и настоящего стандарта.

А.7.1 ApplicationReference (ссылка на приложение)

В А.7.1 определены параметры **ApplicationReference** и **Authentication-value**, которые используются как поля объекта прикладной ассоциации.

```
MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4)}
DEFINITIONS ::= BEGIN
EXPORTS
        ApplicationReference,
        Authentication-value;
IMPORTS
        AP-title,
        AP-invocation-identifier,
        AE-qualifier,
        AE-invocation-identifier,
        Authentication-value
        FROM ACSE-1
            { joint-iso-itu-t association-control(2) modules(0) apdus(0) version1(1)};
    ApplicationReference ::= SEQUENCE {
            ap-title [0] ACSE-1.AP-title OPTIONAL,
            ap-invocation-id [1] ACSE-1.AP-invocation-identifier OPTIONAL,
            ae-qualifier [2] ACSE-1.AE-qualifier OPTIONAL,
            ae-invocation-id [3] ACSE-1.AE-invocation-identifier OPTIONAL
            }
END
```

Параметр **ApplicationReference** выводится в соответствии с правилами настоящего стандарта (см. 5.5). Используется определение параметра услуги ссылки на приложение, данное в разделе 6 ИСО 9506-1. Рассматриваемое определение ASN.1 использует типы **AP-title**, **AP-Invocation-id**, **AE-qualifier** и **AE-Invocation-id**, установленные определением модуля ACSE-1 в соответствии с ИСО/МЭК 8650.

Значения, используемые сущностью **ApplicationReference** в любой реализации, заданы таким образом, чтобы ссылка **ApplicationReference** была достаточной для уникальной и однозначной идентификации прикладного процесса, идентификации порядка активизации прикладного процесса, идентификации сущности приложения (порядка активизации сущности приложения) в соответствии с требованиями ссылочной MMS-услуги.

Примечание — Дополнительную информацию об именовании и адресации прикладных уровней можно найти в ИСО 7498-3, ИСО/МЭК 9545, ИСО/МЭК 8649 и ИСО/МЭК 8650.

Приложение В
(обязательное)

Абстрактный формат конфигурации и инициализации

Настоящее приложение устанавливает правила кодирования информации о конфигурации и инициализации (см. раздел 25). Данные коды использованы в отдельном модуле ASN.1. Этот модуль применяют для передачи информации (см. раздел 25) или для хранения данных инициализации систем.

```
MMS-SCI-Module-1 { iso standard 9506 part(2) mms-file-record-version1(2) }
DEFINITIONS ::= BEGIN
  IMPORTS ApplicationReference
      FROM MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1(4) }
      AccessCondition,
      AdditionalCBBOptions,
      AdditionalSupportOptions,
      Address,
      AlarmAckRule,
      Control-State,
      DomainState,
      EC-Class,
      EC-State,
      EE-Duration,
      EE-Class,
      LogicalStatus,
      Modifier,
      ParameterSupportOptions,
      PhysicalStatus,
      Priority,
      ProgramInvocationState,
      ServiceSupportOptions,
      Severity,
      Transitions,
      TypeDescription
      FROM MMS-Object-Module-1 { iso standard 9506 part(1) mms-object-model1(1) }
      AlternateAccess,
      ConfirmedServiceRequest,
      AttachToEventCondition,
      AttachToSemaphore,
      Data,
      EE-State,
      Identifier,
      Integer8,
      Integer32,
      MMSString,
      MMS255String,
      ObjectName,
      TimeOfDay,
      TypeSpecification,
      Unsigned32,
      Unsigned8
      FROM ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) };
  SCI-Information ::= SEQUENCE {
      partOne                    [0] IMPLICIT VMD-File,
      partTwo                    [1] IMPLICIT Service-and-Parameter-CBBs
      }
```

B.1 SCI. Часть 1: инициализация VMD (виртуального производственного устройства)

Коды информации, содержащейся в таблице 1, соответствуют типу **VMD-File**.

```
VMD-File ::= SEQUENCE {
```

```
        executiveFunction
            [0] IMPLICIT ApplicationReference,
        vendorName
            [1] MMSString,
        modelName
            [2] MMSString,
        revision
            [3] MMSString,
        abstractSyntaxes
            [4] IMPLICIT OBJECT IDENTIFIER,
        -- no TRANSACTIONs,
        -- no APPLICATION-ASSOCIATIONs,
        accessControl
            [5] IMPLICIT Access-Control-List-instance,
        logicalStatus
            [6] IMPLICIT LogicalStatus,
        capabilities
            [7] IMPLICIT SEQUENCE OF MMSString,
        physicalStatus
            [8] IMPLICIT PhysicalStatus,
        local-detail
            [9] IMPLICIT BIT STRING,
        accessControlLists
            [10] IMPLICIT SEQUENCE OF Access-Control-List-instance,
        domains
            [11] IMPLICIT SEQUENCE OF Domain-instance,
        programInvocations
            [12] IMPLICIT SEQUENCE OF Program-Invocation-instance,
        unitControls
            [13] IMPLICIT SEQUENCE OF Unit-Control-instance
IF (vadr)
,       unnamedVariables
            [14] IMPLICIT SEQUENCE OF Unnamed-Variable-instance
ELSE
,       unnamedVariables
            [14] IMPLICIT NULL
ENDIF
IF (vnam)
,       namedVariables
            [15] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,       namedVariableLists
            [16] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,       namedVariableLists
            [16] IMPLICIT NULL
ENDIF
,       namedTypes
            [17] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,       namedVariables
            [15] IMPLICIT NULL,
        namedVariableLists
            [16] IMPLICIT NULL,
        namedTypes
            [17] IMPLICIT NULL
ENDIF
,       dataExchanges
            [18] IMPLICIT SEQUENCE OF Data-Exchange-instance,
        semaphores
```

```
            [19] IMPLICIT SEQUENCE OF Semaphore-instance,
        operatorStations
            [20] IMPLICIT SEQUENCE OF Operator-Station-instance,
        eventConditions
            [21] IMPLICIT SEQUENCE OF Event-Condition-instance,
        eventActions
            [22] IMPLICIT SEQUENCE OF Event-Action-instance,
        eventEnrollments
            [23] IMPLICIT SEQUENCE OF Event-Enrollment-instance
IF (cspi)
,       eventConditionLists
            [24] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ELSE
,       eventConditionLists
            [24] IMPLICIT NULL
ENDIF
,       journals
            [25] IMPLICIT SEQUENCE OF Journal-instance,
        ...
IF (csr)
,       selected-Program-InvocationCHOICE {
        selectedProgram
            [26] IMPLICIT Program-Invocation-instance,
        noneSelected
            [27] IMPLICIT NULL }
ENDIF
        }
```

B.1.1 Объекты перечня средств управления доступом

Ниже представлен пример кодирования информации, содержащейся в таблице 3.

```
    Access-Control-List-instance ::= SEQUENCE {
            name                    [0] IMPLICIT Identifier,
            definition                      CHOICE {
                reference                   [1] IMPLICIT OBJECT IDENTIFIER,
                details                             [2] IMPLICIT SEQUENCE {
                    accessControl
                        [3] IMPLICIT Access-Control-List-instance,
                    readAccessCondition
                        [4] AccessCondition OPTIONAL,
                    storeAccessCondition
                        [5] AccessCondition OPTIONAL,
                    writeAccessCondition
                        [6] AccessCondition OPTIONAL,
                    loadAccessCondition
                        [7] AccessCondition OPTIONAL,
                    executeAccessCondition
                        [8] AccessCondition OPTIONAL,
                    deleteAccessCondition
                        [9] AccessCondition OPTIONAL,
                    editAccessCondition
                        [10] AccessCondition OPTIONAL,
--
-- The following fields are used to record lists of objects placed
-- under the control of this ACCESS-CONTROL-LIST object.
-- They will be referred to collectively as the Controlled Object Lists
--
                    accessControlLists
                        [11] IMPLICIT SEQUENCE OF Access-Control-List-instance,
                    domains
                        [12] IMPLICIT SEQUENCE OF Domain-instance,
```

```
                programInvocations
                    [13] IMPLICIT SEQUENCE OF Program-Invocation-instance,
                unitControls
                    [14] IMPLICIT SEQUENCE OF Unit-Control-instance
IF (vadr)
,               unnamedVariables
                    [15] IMPLICIT SEQUENCE OF Unnamed-Variable-instance
ELSE
,               unnamedVariables
                    [15] IMPLICIT NULL
ENDIF
IF (vnam)
,               namedVariables
                    [16] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,               namedVariableLists
                    [17] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,               namedVariableLists
                    [17] IMPLICIT NULL
ENDIF
,               namedTypes
                    [18] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,               namedVariables
                    [16] IMPLICIT NULL,
                namedVariableLists
                    [17] IMPLICIT NULL,
                namedTypes
                    [18] IMPLICIT NULL
ENDIF
,               dataExchanges
                    [19] IMPLICIT SEQUENCE OF Data-Exchange-instance,
                semaphores
                    [20] IMPLICIT SEQUENCE OF Semaphore-instance,
                operatorStations
                    [21] IMPLICIT SEQUENCE OF Operator-Station-instance,
                eventConditions
                    [22] IMPLICIT SEQUENCE OF Event-Condition-instance,
                eventActions
                    [23] IMPLICIT SEQUENCE OF Event-Action-instance,
                eventEnrollments
                    [24] IMPLICIT SEQUENCE OF Event-Enrollment-instance,
                journals
                    [25] IMPLICIT SEQUENCE OF Journal-instance,
                ...
IF (cspi)
,               eventConditionLists
                    [26] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
                } } }
```

B.1.2 Объекты области

Ниже представлен пример кодирования информации, содержащейся в таблице 4.

```
Domain-instance ::= SEQUENCE {
    name            [0] IMPLICIT Identifier,
    definition  CHOICE {
        reference           [1] IMPLICIT OBJECT IDENTIFIER,
        details         [2] IMPLICIT SEQUENCE {
            capabilities        [3] IMPLICIT SEQUENCE OF MMSString,
```

```
                state                           [4] IMPLICIT DomainState,
    -- The aAssociation is not included
                accessControl
                    [5] IMPLICIT Access-Control-List-instance,
        sharable                    [6] IMPLICIT BOOLEAN,
        programInvocations
                    [7] IMPLICIT SEQUENCE OF Program-Invocation-instance
    -- uploadsInProgress is not included
IF (vnam)
,       namedVariables
                    [8] IMPLICIT SEQUENCE OF Named-Variable-instance
IF (vlis)
,       namedVariableLists
                    [9] IMPLICIT SEQUENCE OF Named-Variable-List-instance
ELSE
,       namedVariableLists
                    [9] IMPLICIT NULL
ENDIF
,       namedTypes
                    [10] IMPLICIT SEQUENCE OF Named-Type-instance
ELSE
,       namedVariables
                    [8] IMPLICIT NULL,
        namedVariableLists
                    [9] IMPLICIT NULL,
        namedTypes
                    [10] IMPLICIT NULL
ENDIF
,       eventConditions
                    [11] IMPLICIT SEQUENCE OF Event-Condition-instance,
        eventActions
                    [12] IMPLICIT SEQUENCE OF Event-Action-instance,
        eventEnrollments
                    [13] IMPLICIT SEQUENCE OF Event-Enrollment-instance
IF (cspi)
,       eventConditionLists
                    [14] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
          }   }   }
```

B.1.3 Объекты предварительно определенного порядка вызова программы

Ниже представлен пример кодирования информации, содержащейся в таблице 5.

```
    Program-Invocation-instance ::= SEQUENCE {
        name                [0] IMPLICIT Identifier,
        definition   CHOICE {
            reference               [1] IMPLICIT OBJECT IDENTIFIER,
            details                 [2] IMPLICIT SEQUENCE {
                programInvocationState
                    [3] IMPLICIT ProgramInvocationState,
                domains
                    [4] IMPLICIT SEQUENCE OF Domain-instance,
                accessControl
                    [5] IMPLICIT SEQUENCE OF Access-Control-List-instance,
                reusable
                    [6] IMPLICIT BOOLEAN,
                monitor
                    [7] IMPLICIT BOOLEAN,
    -- The following three fields shall all be present if the value of
    -- monitor is true.
    -- If present, the &name field of each object instance
```

```
    -- shall have a value equal to the
    -- &name field of this instance of the PROGRAM-INVOCATION.
            eventCondition
                [8] IMPLICIT SEQUENCE OF Event-Condition-instance OPTIONAL,
            eventAction
                [9] IMPLICIT SEQUENCE OF Event-Action-instance OPTIONAL,
            eventEnrollment
                [10] IMPLICIT SEQUENCE OF Event-Enrollment-instance OPTIONAL,
            executionArgument
                [11] MMSString,
            ...
IF (csr)
,           control
                [12] IMPLICIT Control-State,
            controlling-Program-Invocation
                [13] IMPLICIT Program-Invocation-instance,
    -- The following field shall be present
    -- if and only if the value of the &control field is controlling.
            controlled-Program-Invocations
                [14] IMPLICIT SEQUENCE OF Program-Invocation-instance OPTIONAL
ENDIF
        }   }   }
```

B.1.4 Объекты предварительно определенного блока управления

Ниже представлен пример кодирования информации, содержащейся в таблице 6.

```
    Unit-Control-instance ::= SEQUENCE {
        name                [0] IMPLICIT Identifier,
        definition   CHOICE {
            reference               [1] IMPLICIT OBJECT IDENTIFIER,
            details                 [2] IMPLICIT SEQUENCE {
                accessControl
,                   [3] IMPLICIT Access-Control-List-instance,
                domains
                    [4] IMPLICIT SEQUENCE OF Domain-instance,
                programInvocations
                    [5] IMPLICIT SEQUENCE OF Program-Invocation-instance
                }   }   }
```

B.1.5 Объекты непоименованной переменной

Ниже представлен пример кодирования информации, содержащейся в таблице 7.

```
    Unnamed-Variable-instance ::= SEQUENCE {
        address                 [0] Address,
        accessControl           [1] IMPLICIT Access-Control-List-instance,
        typeDescription         [2] TypeDescription
        }
```

B.1.6 Объекты предварительно определенной поименованной переменной

Ниже представлен пример кодирования информации, содержащейся в таблице 8.

```
    Named-Variable-instance ::= SEQUENCE {
        name                [0] ObjectName,
        definition   CHOICE {
            reference               [1] IMPLICIT OBJECT IDENTIFIER,
            details         [2] IMPLICIT SEQUENCE {
                accessControl
                    [3] IMPLICIT Access-Control-List-instance,
                typeDescription                 [4] TypeDescription
IF ( vadr )
,               address                         [5] Address OPTIONAL
ELSE
,                                               [5] NULL
```

```
ENDIF
IF ( sem )
,           meaning                         [6] IMPLICIT VisibleString OPTIONAL
ENDIF
        }       }       }
```

В.1.7 Объекты предварительно определенного перечня поименованных переменных

Ниже представлен пример кодирования информации, содержащейся в таблице 9.

```
    Named-Variable-List-instance ::= SEQUENCE {
        name            [0] ObjectName,
        definition  CHOICE {
            reference           [1] IMPLICIT OBJECT IDENTIFIER,
            details             [2] IMPLICIT SEQUENCE {
                accessControl
                    [3] IMPLICIT Access-Control-List-instance,
                listOfVariables
                    [4] IMPLICIT SEQUENCE OF Variable-List-Item-instance
                }           }       }
    Variable-List-Item-instance ::= SEQUENCE {
    -- one and only one of the following two lines shall appear
IF ( vadr )
        unnamedItem             [0] IMPLICIT Unnamed-Variable-instance OPTIONAL
ELSE
        unnamedItem             [0] IMPLICIT NULL OPTIONAL
ENDIF
IF ( vnam )
,       namedItem               [1] IMPLICIT Named-Variable-instance OPTIONAL
ELSE
,       namedItem               [1] IMPLICIT NULL OPTIONAL
ENDIF
IF (valt)
    -- the following specification may be included
,   alternateAccess             [2] IMPLICIT AlternateAccess OPTIONAL
ENDIF
    }
```

В.1.8 Объекты предварительно определенного поименованного типа

Ниже представлен пример кодирования информации, содержащейся в таблице 10.

```
    Named-Type-instance ::= SEQUENCE {
        name                    [0] ObjectName,
        definition          CHOICE {
            reference                   [1] IMPLICIT OBJECT IDENTIFIER,
            details                 [2] IMPLICIT SEQUENCE {
                accessControl               [3] IMPLICIT Access-Control-List-instance,
                typeDescription     [4] TypeDescription
IF (sem)
,               meaning                     [5] IMPLICIT VisibleString OPTIONAL
ENDIF
                }   }   }
```

В.1.9 Объекты предварительно определенного обмена данными

Ниже представлен пример кодирования информации, содержащейся в таблице 11.

```
    Data-Exchange-instance ::= SEQUENCE {
        name                    [0] IMPLICIT Identifier,
        definition          CHOICE {
            reference                   [1] IMPLICIT OBJECT IDENTIFIER,
            details                 [2] IMPLICIT SEQUENCE {
                accessControl                   [3] IMPLICIT Access-Control-List-instance,
                request                         [4] IMPLICIT SEQUENCE OF TypeDescription,
```

```
        response                    [5] IMPLICIT SEQUENCE OF TypeDescription,
        linked                      [6] IMPLICIT BOOLEAN,
-- The following attribute shall appear if an only if the value of &linked is true.
        programInvocation           [7] IMPLICIT Program-Invocation-instance OPTIONAL
        } } }
```

B.1.10 Объекты предварительно определенного семафора

Ниже представлен пример кодирования информации, содержащейся в таблице 12.

```
Semaphore-instance ::= SEQUENCE {
    name                    [0] IMPLICIT Identifier,
    definition          CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl           [3] IMPLICIT Access-Control-List-instance,
            class                   [4] IMPLICIT ENUMERATED {
                token,
                pool },
-- If the value of &class is token, the following field shall appear
            numberOfTokens      [5] IMPLICIT INTEGER OPTIONAL,
-- If the value of &class is pool, the following field shall appear
            namedTokens             [6] IMPLICIT SEQUENCE OF VisibleString OPTIONAL,
            eventCondition          [7] IMPLICIT Event-Condition-instance
            } } }
```

B.1.11 Объекты предварительно определенной станции управления

Ниже представлен пример кодирования информации, содержащейся в таблице 13.

```
Operator-Station-instance ::= SEQUENCE {
    name                    [0] IMPLICIT Identifier,
    definition          CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl           [3] IMPLICIT Access-Control-List-instance,
            stationType         [4] IMPLICIT ENUMERATED {
                entry,
                display,
                entry-display }
            } } }
```

B.1.12 Объекты предварительно определенных условий события

Ниже представлен пример кодирования информации, содержащейся в таблице 14.

```
Event-Condition-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition          CHOICE {
        reference               [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl           [3] IMPLICIT Access-Control-List-instance,
            ecClass                 [4] IMPLICIT EC-Class,
            ecState                 [5] IMPLICIT EC-State,
            priority                [6] IMPLICIT Priority,
            severity                [7] IMPLICIT Severity,
            eventEnrollments        [8] IMPLICIT SEQUENCE OF Event-Enrollment-instance,
-- The following fields shall be present
-- if and only if the value of &ecClass is monitored.
            enabled                 [9] IMPLICIT BOOLEAN OPTIONAL,
            alarmSummaryReports     [10] IMPLICIT BOOLEAN OPTIONAL,
            monitoredVariable       CHOICE {
                named                   [11] IMPLICIT Named-Variable-instance,
                unnamed                     [12] IMPLICIT Unnamed-Variable-instance,
```

```
                unspecified                                 [13] IMPLICIT NULL } OPTIONAL,
            evaluationInterval          [14] IMPLICIT INTEGER OPTIONAL,
            ...
IF (cspi)
,           displayEnhancement          CHOICE {
IF (des)
                text                                        [15] MMSString
ENDIF
IF (dei)
,               number                          [16] IMPLICIT INTEGER
ENDIF
,               none                                        [17] IMPLICIT NULL
                },
            group-Priority-Override     CHOICE {
                priority                        [18] IMPLICIT Priority,
                undefined                                   [19] IMPLICIT NULL
                } OPTIONAL,
            referencingEventConditionLists
                    [20] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
            }   }   }
```

B.1.13 Объекты предварительно определенных действий события

Ниже представлен пример кодирования информации, содержащейся в таблице 15.

```
Event-Action-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition          CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
        details                     [2] IMPLICIT SEQUENCE {
            accessControl                           [3] IMPLICIT Access-Control-List-instance,
            confirmedServiceRequest                 [4] ConfirmedServiceRequest,
            modifiers                               [5] IMPLICIT SEQUENCE OF Modifier,
            eventEnrollments                [6] IMPLICIT SEQUENCE OF Event-Enrollment-instance
            }   }   }
```

B.1.14 Объекты предварительно определенной регистрации события

Ниже представлен пример кодирования информации, содержащейся в таблице 16.

```
Event-Enrollment-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition          CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
        details                 [2] IMPLICIT SEQUENCE {
            accessControl               [3] IMPLICIT Access-Control-List-instance,
            eeClass                     [4] IMPLICIT EE-Class ,
            eventCondition              [5] IMPLICIT Event-Condition-instance,
            ecTransitions               [6] IMPLICIT Transitions,
-- The following parameter is present if and only if the
-- value of &eeClass is modifier.
            remainingDelay          CHOICE {
                time                                    [7] IMPLICIT INTEGER,
                forever                         [8] IMPLICIT NULL } OPTIONAL,
-- The remaining parameters are present if and only if the
-- value of &eeClass is notification.
            eventAction                 [9] IMPLICIT Event-Action-instance OPTIONAL,
            duration                                    [10] IMPLICIT EE-Duration OPTIONAL,
            clientApplication                           [11] IMPLICIT ApplicationReference OPTIONAL,
            aaRule                                      [12] IMPLICIT AlarmAckRule OPTIONAL,
            ...
IF (cspi)
,           displayEnhancement          CHOICE {
IF (des)
```

```
                text                                    [13] MMSString
ENDIF
IF (dei)
,               number                          [14] IMPLICIT INTEGER
ENDIF
,               none                                    [15] IMPLICIT NULL
}
ENDIF
            }   }   }
```

B.1.15 Объекты предварительно определенного перечня условий события

Ниже представлен пример кодирования информации, содержащейся в таблице 17.

```
Event-Condition-List-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition          CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl
                [3] IMPLICIT Access-Control-List-instance,
            eventConditions
                [4] IMPLICIT SEQUENCE OF Event-Condition-instance
IF (recl)
,           eventConditionLists
                [5] IMPLICIT SEQUENCE OF Event-Condition-List-instance,
            referencingEventConditionLists
                [6] IMPLICIT SEQUENCE OF Event-Condition-List-instance
ENDIF
                }   }   }
```

B.1.16 Объекты предварительно определенного журнала

Ниже представлен пример кодирования информации, содержащейся в таблице 18.

```
Journal-instance ::= SEQUENCE {
    name                    [0] ObjectName,
    definition          CHOICE {
        reference                   [1] IMPLICIT OBJECT IDENTIFIER,
        details             [2] IMPLICIT SEQUENCE {
            accessControl                   [3] IMPLICIT Access-Control-List-instance,
            entries                         [4] IMPLICIT SEQUENCE OF Journal-Entry-instance
            }   }   }
Journal-Entry-instance ::= SEQUENCE {
    journal                         [0] IMPLICIT Journal-instance,
    entry                           [1] IMPLICIT OCTET STRING,
    clientApplication       [2] IMPLICIT ApplicationReference,
    timeStamp                       [3] IMPLICIT TimeOfDay,
    orderOfReceipt          [4] IMPLICIT INTEGER,
    informationType         [5] IMPLICIT ENUMERATED {
        annotation,
        event-data,
        data },
-- The following attribute shall appear if and only if the
-- value of &informationType is annotation.
    textComment                     [6] MMS255String OPTIONAL,
--The following attribute shall appear if and only if the
-- value of &informationType is event-data.
    eventTransitionsRecord  [7] IMPLICIT SEQUENCE {
        name                        [8] ObjectName,
        currentState                [9] IMPLICIT EC-State
        } OPTIONAL,
-- The following attribute shall appear if and only if the
-- value of &informationType is data or event-data.
```

```
        journalVariables            [10] IMPLICIT SEQUENCE OF SEQUENCE {
                variableTag                     [11] MMS255String,
                valueSpecification              [12] Data
                } OPTIONAL
        }
```

B.2 Услуги и параметры СВВ

Спецификация информации, содержащейся в SCI часть 2, таблицы 20–52, описана типом **Service-and-Parameter-CBBs**:

```
    Service-and-Parameter-CBBs ::= SEQUENCE {
        services-Client             [0] IMPLICIT ServiceSupportOptions,
        services-Server             [1] IMPLICIT ServiceSupportOptions,
        parameters                      [2] IMPLICIT ParameterSupportOptions,
        nest                            [3] IMPLICIT INTEGER
IF (csr cspi)
,       extendedServices-Client     [4] IMPLICIT AdditionalSupportOptions,
        extendedServices-Server     [5] IMPLICIT AdditionalSupportOptions
ELSE
,       extendedServices-Client     [4] IMPLICIT NULL,
        extendedServices-Server     [5] IMPLICIT NULL
ENDIF
IF (cspi)
,       extendedParameters              [6] IMPLICIT AdditionalCBBOptions
ELSE
,       extendedParameters              [6] IMPLICIT NULL
ENDIF
,       generalManagement               [7] IMPLICIT GeneralManagementParameters,
        vMDSupport                      [8] IMPLICIT VMDSupportParameters,
        domainManagement                [9] IMPLICIT DomainManagementParameters,
        programInvocation           [10] IMPLICIT ProgramInvocationManagementParameters,
        variableAccess                  [11] IMPLICIT VariableAccessParameters,
        dataParameters              [12] IMPLICIT DataParameters,
        semaphoreManagement         [13] IMPLICIT SemaphoreManagementParameters,
        operatorCommunication       [14] IMPLICIT OperatorCommunicationParameters,
        errors                          [15] IMPLICIT ErrorParameters,
        fileManagement              [16] IMPLICIT FileManagementParameters
        }
```

B.2.1 Параметры среды и общего управления

Ниже представлен пример кодирования информации, содержащейся в таблице 21.

```
    GeneralManagementParameters ::= SEQUENCE {
        localDetail                     [0] MMSString,
        supportForTime              [1] IMPLICIT SEQUENCE {
            timeOfDay                       [2] IMPLICIT BOOLEAN,
            timeSequence                [3] IMPLICIT BOOLEAN
            },
        granularityOfTime           [4] IMPLICIT INTEGER
        }
```

B.2.2 Параметры поддержки VMD

Ниже представлен пример кодирования информации, содержащейся в таблице 25.

```
    VMDSupportParameters ::= SEQUENCE {
        localDetail                     [0] MMSString,
        extendedDerivation              [1] MMSString
                    -- method used to perform extended derivation
        }
```

B.2.3 Параметры управления доменом

Ниже представлен пример кодирования информации, содержащейся в таблице 27.

```
    DomainManagementParameters ::= SEQUENCE {
```

```
    loadDataOctet                   [0] MMSString,
        -- description of the format of Load Data if the octet string form is used
    loadDataSyntax                  [1] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
        -- identifier of the Abstract Syntaxes used
    maxUploads                              [2] IMPLICIT INTEGER
    }
```

В.2.4 Параметры управления процедурой вызова программы

Ниже представлен пример кодирования информации, содержащейся в таблице 28.

```
ProgramInvocationManagementParameters ::= SEQUENCE {
    executionArgMaxSize                     [0] IMPLICIT INTEGER,
    executionArgParseRules          [1] MMSString,
    executionArgSyntaxes                    [2] IMPLICIT SEQUENCE OF OBJECT IDENTIFIER,
    programLoction                  [3] MMSString,
            -- syntax of the program Location notation
    stepMode                        [4] IMPLICIT BOOLEAN
            -- if true, step Mode is supported
    }
```

В.2.5 Параметры доступа к переменной

Ниже представлен пример кодирования информации, содержащейся в таблице 32.

```
VariableAccessParameters ::= SEQUENCE {
    uninterruptibleAccess           [0] MMSString,
        -- conditions under which it is guaranteed
    singleMode                      [1] IMPLICIT BOOLEAN,
    unnamedMode                     [2] IMPLICIT BOOLEAN
    }
```

В.2.6 Параметры доступа к переменной

Ниже представлен пример кодирования информации, содержащейся в таблице 33.

```
DataParameters ::= SEQUENCE {
    bit-string                              [0] IMPLICIT INTEGER OPTIONAL,
    integer                                 [1] IMPLICIT INTEGER OPTIONAL,
    unsigned                                [2] IMPLICIT INTEGER OPTIONAL,
    floating-point                          [3] IMPLICIT SEQUENCE {
        total                               [4] IMPLICIT INTEGER,
        exponent                            [5] IMPLICIT INTEGER } OPTIONAL,
    octet-string                            [10] IMPLICIT INTEGER OPTIONAL,
    visible-string                          [11] IMPLICIT INTEGER OPTIONAL,
    binary-time                             [12] IMPLICIT BOOLEAN OPTIONAL,
    bcd                                     [13] IMPLICIT INTEGER OPTIONAL,
    mmsString                               [14] IMPLICIT INTEGER OPTIONAL
    }
```

В.2.7 Параметры управления семафором

Ниже представлен пример кодирования информации, содержащейся в таблице 36.

```
SemaphoreManagementParameters ::= SEQUENCE {
    algorithm                   [0] IMPLICIT MMSString
        -- method of processing the &priority field
    }
```

В.2.8 Параметры связи с оператором

Ниже представлен пример кодирования информации, содержащейся в таблице 37.

```
OperatorCommunicationParameters ::= SEQUENCE {
    input-time-out                  [0] IMPLICIT INTEGER
    }
```

В.2.9 Параметры ошибки

Ниже представлен пример кодирования информации, содержащейся в таблице 47.

```
ErrorParameters ::= SEQUENCE {
```

```
    additionalCode                  [0] MMSString,
    additionalDetial            [1] IMPLICIT SEQUENCE {
        size                                [2] IMPLICIT INTEGER,
        syntax                      [3] MMSString
        }
    }
```

B.2.10 Параметры управления файлом

Ниже представлен пример кодирования информации, содержащейся в таблице 50.

```
FileManagementParameters ::= SEQUENCE {
    fileName                    [0] MMSString
    }
END
```

Приложение С
(обязательное)

Протокол доступа к файлу

С.1 Введение

Настоящий раздел содержит описания особых элементов протокола услуги, описанной в приложении С (услуга доступа к файлу), содержащем определение MMS-услуги (см. ИСО 9506-1). Здесь описана только услуга **ObtainFile**. Ниже представлен модуль, определенный в настоящем приложении и в приложениях D и E.

```
    ISO-9506-MMS-1A { iso standard 9506 part(2) mms-annex-version1(3) }
DEFINITIONS ::= BEGIN
EXPORTS
        ObtainFile-Request,
        ObtainFile-Response,
        ObtainFile-Error,
        FileOpen-Request,
        FileOpen-Response,
        FileRead-Request,
        FileRead-Response,
        FileClose-Request,
        FileClose-Response,
        FileRename-Request,
        FileRename-Response,
        FileRename-Error,
        FileDelete-Request,
        FileDelete-Response,
        FileDirectory-Request,
        FileDirectory-Response,
        ScatteredAccessDescription,
        DefineScatteredAccess-Request,
        DefineScatteredAccess-Response,
        GetScatteredAccessAttributes-Request,
        GetScatteredAccessAttributes-Response;
IMPORTS
        FileName,
        ObjectName,
        AlternateAccess,
        VariableSpecification,
        Identifier,
        Integer32,
        Unsigned32 FROM
        ISO-9506-MMS-1 { iso standard 9506 part(2) mms-abstract-syntax-version1(1) }
        ApplicationReference FROM
        MMS-Environment-1 { iso standard 9506 part(2) mms-environment-version1 (4) };
```

С.2 Услуга ObtainFile (получить файл)

Абстрактный синтаксис выбора **obtainFile** типов запроса подтверждаемой услуги, ответа услуги и ошибки услуги описан ниже. В 5.5 установлен порядок вывода всех параметров, описание которых отсутствует в настоящем подразделе.

```
    ObtainFile-Request ::= SEQUENCE {
IF ( tpy )
        sourceFileServer            [0] IMPLICIT ApplicationReference OPTIONAL,
ENDIF
        sourceFile                      [1] IMPLICIT FileName,
        destinationFile                 [2] IMPLICIT FileName
        }
    ObtainFile-Response ::= NULL
    ObtainFile-Error ::= INTEGER {
        source-file                     (0),
```

destination-file (1)
} (0..1)

C.2.1 ObtainFile-Request (запрос получения файла)

Абстрактный синтаксис выбора **ObtainFile** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **ObtainFile-Request**.

C.2.2 ObtainFile-Response (ответ получения файла)

Абстрактный синтаксис выбора **ObtainFile** выбор для ответа услуги **ServiceResponce** — это тип **ObtainFile-Response**.

C.2.3 ObtainFile-Error (ошибка получения файла)

Абстрактный синтаксис выбора **ObtainFile** для выбора **ServiceSpecificInformation** типа подтверждаемой услуги **ConfirmedServiceError** — это ошибка получения файла **ObtainFile-Error.** Данная сущность является подпараметром **File in Error** параметра **Result(–)** примитива ответа **ObtainFile.response.** Она выглядит как подпараметр **File in Error** параметра **Result(–)** примитива подтверждения **ObtainFile.confirm** (при наличии).

Приложение D
(справочное)

Протокол управления файлом

Примечание — Настоящее приложение не является нормативным. Однако для обеспечения соответствия требованиям корректной работы с протоколами в настоящем приложении предпочтение отдается не рекомендациям, а нормативному языку.

D.1 Overview (обзор)

Настоящий подраздел содержит описания особых элементов протокола услуг, определяемых приложением С ИСО 9506-1. Раздел содержит описания следующих услуг:

FileOpen
FileRead
FileCLose
FileRename
FileDelete
FileDirectory

D.2 FileOpen (открыть файл)

Абстрактный синтаксис выбора **FileOpen** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileOpen-Request ::= SEQUENCE {
    fileName            [0] IMPLICIT FileName,
    initialPosition             [1] IMPLICIT Unsigned32 }
FileOpen-Response ::= SEQUENCE {
    frsmID                      [0] IMPLICIT Integer32,
    fileAttributes              [1] IMPLICIT FileAttributes }
```

D.2.1 FileOpen-Request (запрос открытия файла)

Абстрактный синтаксис выбора **FileOpen** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **FileOpen-Request**.

D.2.2 FileOpen-Response (ответ открытия файла)

Абстрактный синтаксис выбора **FileOpen** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это **FileOpen-Response**.

D.3 FileRead (читать файл)

Абстрактный синтаксис выбора **FileRead** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileRead-Request ::= Integer32 -- FRSM ID
FileRead-Response ::= SEQUENCE {
    fileData                    [0] IMPLICIT OCTET STRING,
    moreFollows                 [1] IMPLICIT BOOLEAN DEFAULT TRUE }
```

D.3.1 FileRead-Request (запрос чтения файла)

Абстрактный синтаксис выбора **FileRead** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это запрос **FileRead-Request**. Это параметр **FRSM ID** запроса чтения файла **FileRead.request.** Он выглядит как параметр **FRSM ID** примитива отображения **FileRead.indication**.

D.3.2 FileRead-Response (ответ чтения файла)

Абстрактный синтаксис выбора **FileRead** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это ответ **FileRead-Response**.

D.4 FileClose (закрыть файл)

Абстрактный синтаксис выбора **FileCLose** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileClose-Request ::= Integer32 – FRSM ID
FileClose-Response ::= NULL
```

D.4.1 FileClose-Request (запрос открытия файла)

Абстрактный синтаксис выбора **FileCLose** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это запрос **FileCLose-Request**. Это параметр **FRSM ID** запроса **FileCLose.request**. Он выглядит как параметр **FRSM ID** примитива отображения **FileCLose.indication**.

D.4.2 FileClose-Response (ответ закрытия файла)

Абстрактный синтаксис выбора **FileCLose** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это ответ **FileCLose-Response**, имеющий тип **NULL.** Данный тип отображается величиной параметра **Result(+)** примитива ответа **FileCLose.response**. Данный параметр выглядит как параметр **Result(+)** примитива подтверждения **FileCLose.confirm**.

D.5 FileRename (переименовать файл)

Абстрактный синтаксис выбора **FileRename** запроса подтверждаемой услуги **ConfirmedServiceRequest**, ответа подтверждаемой услуги **ConfirmedServiceResponse** и ошибки подтверждаемой услуги **ConfirmedServiceError** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileRename-Request ::= SEQUENCE {
    currentFileName          [0] IMPLICIT FileName,
    newFileName                  [1] IMPLICIT FileName }
FileRename-Response ::= NULL
FileRename-Error ::= INTEGER {
    source-file                  (0),
    destination-file             (1)
    } (0..1)
```

D.5.1 FileRename-Request (запрос переименования файла)

Абстрактный синтаксис выбора **FileRename** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это **FileRename-Request**.

D.5.2 FileRename-Response (ответ переименования файла)

Абстрактный синтаксис выбора **FileRename** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это ответ **FileRename-Response**. Данный тип отображается величиной параметра **Result(+)** примитива ответа **FileRename.response**. Он выглядит как параметр **Result(+)** примитива подтверждения **FileRename. confirm**.

D.5.3 FileRename-Error (ошибка переименования файла)

Абстрактный синтаксис выбора **FileRename** для типа **ConfirmedServiceError** — это ошибка **FileRename-Error**, которая является подпараметром **File In Error** параметра **Result(–)** примитива ответа **FileRename.response**. Он выглядит как подпараметр **File In Error** параметра **Result(–)** примитива подтверждения **FileRename.confirm** (при его наличии).

D.6 FileDelete (стереть файл)

Абстрактный синтаксис выбора **FileDelete** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileDelete-Request ::= FileName
FileDelete-Response ::= NULL
```

D.6.1 FileDelete-Request (запрос удаления файла)

Абстрактный синтаксис выбора **FileDelete** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это запрос **FileDelete-Request**. Это параметр **File Name** примитива запроса **FileDelete.request**. Он выглядит как параметр **File Name** примитива отображения **FileDelete.indication**.

D.6.2 FileDelete-Response (ответ удаления файла)

Абстрактный синтаксис выбора **FileDelete** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это ответ **FileDelete-Response**. Данный тип отображается величиной параметра **Result(+)** примитива ответа **FileDelete. response**. Он выглядит как параметр **Result(+)** примитива подтверждения **FileDelete.confirm**.

D.7 FileDirectory (директория файла)

Абстрактный синтаксис выбора **FileDirectory** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
FileDirectory-Request ::= SEQUENCE {
    fileSpecification           [0] IMPLICIT FileName OPTIONAL,
    continueAfter                   [1] IMPLICIT FileName OPTIONAL }
FileDirectory-Response ::= SEQUENCE {
    listOfDirectoryEntry            [0] SEQUENCE OF DirectoryEntry,
    moreFollows                     [1] IMPLICIT BOOLEAN DEFAULT FALSE }
DirectoryEntry ::= SEQUENCE {
    fileName                [0] IMPLICIT FileName,
    fileAttributes                  [1] IMPLICIT FileAttributes }
```

D.7.1 FileDirectory-Request (запрос директории файла)

Абстрактный синтаксис выбора **FileDirectory** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это запрос **FileDirectory-Request**.

D.7.2 FileDirectory-Response (ответ директории файла)

Абстрактный синтаксис выбора **FileDirectory** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это ответ **FileDirectory-Response**.

D.7.2.1 ListOfDirectoryEntry (перечень записей директории)

Поле **ListOfDirectoryEntry** — это параметр **List Of Directory Entry** примитива ответа **FileDirectory.response**. Он выглядит как параметр **List Of Directory Entry** примитива подтверждения **FileDirectory.confirm**. Данное поле содержит нуль и более реализаций типа **DirectoryEntry.** Каждая реализация содержит значение одного подпараметра **DirectoryEntry** параметра **List Of Directory Entry**, взятого в указанном порядке. Каждая реализация подпараметра **DirectoryEntry** параметра **List Of Directory Entry** должна соответствовать требованиям 5.5 для корректного получения соответствующего элемента последовательности **ListOfDirectoryEntry**.

D.8 FileAttributes (атрибуты файла)

Абстрактный синтаксис параметра **FileAttributes** описан ниже.

```
FileAttributes ::= SEQUENCE {
    sizeOfFile          [0] IMPLICIT Unsigned32, -- in octets
    lastModified                [1] IMPLICIT GeneralizedTime OPTIONAL }
```

Приложение Е
(справочное)

Рассеянный доступ

Е.1 Введение

Следующие особенности определены ИСО/МЭК 9506. Настоящее приложение является справочным. Однако оно использует нормативный язык для представления текста из первого издания указанного стандарта.

В настоящем разделе приведено описание особых элементов протокола услуг, необходимых для поддержки рассеянного доступа. Данный подраздел включает протокол, необходимый для реализации следующих услуг:

DefineScatteredAccess
GetScatteredAccessAttributes

Е.1.1 Протокол спецификации доступа к переменным

Е.1.2 VariableSpecification (спецификация переменной)

Наличие объектов рассеянного доступа изменяет порядок разработки спецификации переменных **VariableSpecification** (см. 14.5.2). В добавление к имеющимся трем вариантам выбора, представлен и четвертый вариант, представляющий собой услугу описания рассеянного доступа **ScatteredAccessDescription**.

Е.1.3 ScatteredAccessDescription (описание рассеянного доступа)

Абстрактный синтаксис параметра **ScatteredAccessDescription** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем пункте.

```
ScatteredAccessDescription ::=SEQUENCE OF SEQUENCE {
    componentName           [0] IMPLICIT Identifier OPTIONAL,
    variableSpecification           [1] VariableSpecification
IF ( valt )
,   alternateAccess         [2] IMPLICIT AlternateAccess OPTIONAL
ENDIF
    }
```

Е.2 DefineScatteredAccess (определение рассеянного доступа)

Абстрактный синтаксис выбора **defineScatteredAccess** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже. В 5.5 установлен порядок получения всех параметров, не описанных в настоящем подразделе.

```
DefineScatteredAccess-Request ::= SEQUENCE {
    scatteredAccessName         [0] ObjectName,
    scatteredAccessDescription  [1] IMPLICIT ScatteredAccessDescription }
DefineScatteredAccess-Response ::= NULL
```

Е.2.1 DefineScatteredAccess-Request (запрос определения рассеянного доступа)

Абстрактный синтаксис выбора **defineScatteredAccess** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **DefineScatteredAccess-Request.**

Е.2.2 DefineScatteredAccess-Response (ответ определения рассеянного доступа)

Абстрактный синтаксис выбора **defineScatteredAccess** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **DefineScatteredAccess-Response**, соответствующий типу **NULL.**

Е.3 GetScatteredAccessAttributes (получение атрибутов рассеянного доступа)

Абстрактный синтаксис выбора **GetScatteredAccessAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** и ответа подтверждаемой услуги **ConfirmedServiceResponse** описан ниже.

```
GetScatteredAccessAttributes-Request ::= ObjectName – ScatteredAccessName
GetScatteredAccessAttributes-Response ::= SEQUENCE {
    mmsDeletable                [0] IMPLICIT BOOLEAN,
    scatteredAccessDescription  [1] IMPLICIT ScatteredAccessDescription
IF ( aco )
,   accessControlList           [2] IMPLICIT Identifier OPTIONAL
        -- Shall not appear in minor version one or two
```

ENDIF
}

E.3.1 GetScatteredAccessAttributes-Request (запрос получения атрибутов рассеянного доступа)

Абстрактный синтаксис выбора **GetScatteredAccessAttributes** запроса подтверждаемой услуги **ConfirmedServiceRequest** — это тип **GetScatteredAccessAttributes-Request**.

E.3.2 GetScatteredAccessAttributes-Response (ответ получения атрибутов рассеянного доступа)

Абстрактный синтаксис выбора **GetScatteredAccessAttributes** ответа подтверждаемой услуги **ConfirmedServiceResponse** — это тип **GetScatteredAccessAttributes-Response**.

E.3.2.1 Перечень средств управления доступом

Параметр **AccessControlList** появляется только в том случае, если оговорено значение **aco CBB**.

Приложение F
(справочное)

Тип данных REAL

F.1 Введение

Следующие особенности установлены в первом издании ИСО/МЭК 9506. В настоящее время использовать его не рекомендуется. Данные особенности обсуждаются только для соблюдения исторической целостности. Настоящее приложение является справочным. Однако нормативный язык применяют для представления текстов первого издания указанного стандарта.

F.2 Данные типа REAL (действительные)

Тип действительных данных ранее поддерживался параметром **TypeDescription** и параметром **Data.** Указанный тип данных ссылается на тип данных **REAL**, определенный в ИСО/МЭК 8824-1.

F.3 Конец модуля

Нижеследующее утверждение **END** заканчивает рассмотрение модуля, начатое в приложении С.

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 646	–	*
ИСО/МЭК 7498-1	IDT	ГОСТ Р ИСО/МЭК 7498-1–99 «Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая модель»
ИСО 7498-2	IDT	ГОСТ Р ИСО 7498-2–99 «Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 2. Архитектура защиты информации»
ИСО/МЭК 7498-3		ГОСТ Р ИСО 7498-3–97 «Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 3. Присвоение имен и адресация»
ИСО 8571 (все части)	–	*
ИСО/МЭК 8650-1	–	*
ИСО/МЭК 8822	–	*
ИСО/МЭК 8824-1	IDT	ГОСТ Р ИСО/МЭК 8824-1–2001 «Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации»
ИСО/МЭК 8824-2	IDT	ГОСТ Р ИСО/МЭК 8824-2–2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 2. Спецификация информационного объекта
ИСО/МЭК 8825-1	IDT	ГОСТ Р ИСО/МЭК 8825-1–2003 «Информационная технология. Правила кодирования АСН.1. Часть 1. Спецификация базовых (BER), канонических (CER) и отличительных (DER) правил кодирования»
ИСО/МЭК 8825-2	IDT	ГОСТ Р ИСО/МЭК 8825-2–2003 «Информационная технология. Правила кодирования АСН.1. Часть 2. Спецификация правил уплотненного кодирования (PER)»
ИСО 9506-1	–	*
ИСО/МЭК 9545	IDT	ГОСТ Р ИСО/МЭК 9545—98 «Информационная технология. Взаимосвязь открытых систем. Структура прикладного уровня»
ИСО/МЭК 10731	–	*
ANSI/IEEE 754	–	*

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

- IDT — идентичные стандарты.

УДК 681.224:621.391:006.354 ОКС 25.040.40 Т58

Ключевые слова: межсетевое взаимодействие приложения, ссылочная модель OSI, протокол прикладного уровня, система управления процессом, система обработки информации, программируемый контроллер, программируемое устройство, спецификация производственных сообщений, система управления робототехническим оборудованием, система цифрового управления, виртуальное производственное устройство, взаимосвязь открытых систем

Редактор *Л.К. Стегниенко*
Технический редактор *А.Б. Заварзина*
Корректор *В.Г. Смолин*
Компьютерная верстка *Д.Е. Першин*

Сдано в набор 24.09.2015. Подписано в печать 8.10.2015. Формат 60х841/8. Гарнитура Ариал.
Усл. печ. л. 20,46. Уч.-изд. л. 17,40. Тираж 30 экз. Зак. 3357.

Набрано в ООО «Академиздат».
gost.e-ecolog.ru gost.e-ecolog.ru

Издано и отпечатано во
ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru