

ГОСТРИСО/МЭК10165-4—2001

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

**ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
СТРУКТУРА ИНФОРМАЦИИ
АДМИНИСТРАТИВНОГО УПРАВЛЕНИЯ**

Часть 4

Руководство по определению управляемых объектов

Издание официальное

БЗЗ—2001/47

ГОССТАНДАРТРОССИИ
Москва

ГОСТРИСО/МЭК10165-4—2001

Предисловие

1 РАЗРАБОТАН Государственным научно-исследовательским и конструкторско-технологическим институтом «ТЕСТ» Министерства Российской Федерации по связи и информатизации

ВНЕСЕН Министерством Российской Федерации по связи и информатизации

2 ПРИНЯТИ И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 6 сентября 2001 г. № 376-ст

3 Настоящий стандарт содержит полный аутентичный текст международного стандарта ИСО/МЭК 10165-4:1992 «Информационная технология. Взаимосвязь открытых систем. Структура информации административного управления. Часть 4. Руководство по определению управляемых объектов» с учетом Изменения № 1 (1996 г.) и Дополнений № 1 (1996 г.), № 2 (1998 г.), № 3 (1998 г.)

4 ВВЕДЕН В ПЕРВЫЕ

© ИПКИздателъство стандартов, 2001

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Госстандарта России

Содержание

1 Область применения	1
2 Нормативные ссылки	2
3 Определения	2
4 Сокращения	4
5 Соглашения	4
6 Общие вопросы	4
7 Общие принципы определения управляемых объектов	11
8 Обозначения для определений управляемых объектов	17
9 Руководство по разработке эквивалентных модулей АСН.1:1994 и АСН.1:1990	38
10 Соглашения для АСН.1 и директив РОУО	43
Приложение А Примеры	48
Приложение В Руководство по применению Z при формализации поведения управляемых объектов	52

Информационная технология
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
СТРУКТУРА ИНФОРМАЦИИ АДМИНИСТРАТИВНОГО УПРАВЛЕНИЯ

Часть 4

Руководство по определению управляемых объектов

Information technology. Open Systems Interconnection. Structure of management information.
Guidelines for the definition of managed objects

Дата введения 2002—07—01

1 Область применения

Настоящий стандарт является руководством для разработчиков других стандартов и рекомендаций, содержащих определения управляемых объектов, которое:

- а) обеспечивает согласованность между определениями управляемых объектов;
- б) гарантирует разработку названных определений способом, совместимым со стандартами и рекомендациями стандартов по административному управлению ВОС;
- г) уменьшает дублирование усилий различных рабочих групп, идентифицируя общие полезные компоновки документов, процедуры и определения.

В настоящем стандарте определены:

- а) взаимосвязи между стандартами и рекомендациями, относящимися к административному управлению ВОС, и определениями классов управляемых объектов, а также принципы использования этих рекомендаций и стандартов в определениях классов управляемых объектов;
- б) методы, применяемые для определения классов управляемых объектов, их атрибутов, сообщений, действий и поведения, включая:
 - 1) сводку вопросов, которые должны быть отражены в определении,
 - 2) обозначения, которые рекомендуется использовать в определении,
 - 3) руководства по согласованности, которым могут следовать определения;
- г) взаимосвязи между определениями классов управляемых объектов и протоколами административного управления и то, что требуется вносящих их протоколу определения;
- д) рекомендуемая структура документации для определений классов управляемых объектов.

Настоящий стандарт применяется для разработки любых рекомендаций и стандартов, определяющих:

- а) информацию административного управления, которая должна быть передана или обработана с помощью протокола административного управления ВОС;
- б) управляемые объекты, к которым относится эта информация.

В настоящем стандарте не определяются и не подразумеваются:

- а) какие-либо ограничения на разработку определений классов управляемых объектов в терминах их функциональных возможностей, на стандарты и рекомендации, которые к ним относятся, или на их использование в конкретной среде административного управления;
- б) руководство по определению ресурсов; в стандарте приводится только руководство по определению управляемых объектов, которые обеспечивают точку зрения административного управления на ресурсы.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты:

ГОСТРИСО/МЭК 7498-1—99 Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая эталонная модель

ГОСТРИСО/МЭК10165-4—2001

ГОСТРИСО7498-3—97Информационнаятехнология.Взаимосвязьоткрытыхсистем.Базовая эталоннаямодель.Часть3.Присвоениеимениадресация

ГОСТРИСО/МЭК7498-4—99Информационнаятехнология.Взаимосвязьоткрытыхсистем. Базоваяэталоннаямодель.Часть4.Основыадминистративногоуправления

ГОСТРИСО/МЭК8824—93Информационнаятехнология.Взаимосвязьоткрытыхсистем.Спецификацияабстрактно-синтаксическойнотацияверсии1(АСН.1)

ГОСТРИСО/МЭК9595—99Информационнаятехнология.Взаимосвязьоткрытыхсистем. Определениеобщихуслугадминистративногоуправления

ГОСТРИСО/МЭК10040—99Информационнаятехнология.Взаимосвязьоткрытыхсистем. Основныеположенияадминистративногоуправлениясистемы

ГОСТРИСО/МЭК10164-1—99Информационнаятехнология.Взаимосвязьоткрытыхсистем. Административноеуправлениеисистемы.Часть1.Функцииадминистративногоуправленияобъектом

ГОСТРИСО/МЭК10164-2—99Информационнаятехнология.Взаимосвязьоткрытыхсистем. Административноеуправлениеисистемы.Часть2.Функцииадминистративногоуправлениясостоянием

ГОСТРИСО/МЭК10165-1—2001Информационнаятехнология.Взаимосвязьоткрытыхсистем. Структураинформацииадминистративногоуправления.Часть1.Модельинформацииадминистративногоуправления

ГОСТРИСО/МЭК10165-2—2001Информационнаятехнология.Взаимосвязьоткрытыхсистем. Структураинформацииадминистративногоуправления.Часть2.Определениеинформацииадминистративногоуправления

ИСО/МЭК8824-1—98*Информационнаятехнология.Абстрактнаясинтаксическаянотация версии1(АСН.1).Часть1.Спецификацияосновнойнотации

ИСО/МЭК9594-2—98*Информационнаятехнология.Взаимосвязьоткрытыхсистем.Справочник.Часть2.Общееописаниепринципов,моделейиуслуг

ИСО/МЭК9596-1—98Информационнаятехнология.Взаимосвязьоткрытыхсистем.Протокол информацииадминистративногоуправления.Часть1.Спецификацияпротокола

ИСО/МЭК9834-1—93*Информационнаятехнология.Взаимосвязьоткрытыхсистем.ПроцедурыработыполномочныхоргановрегистрацииВОС.Часть1.Общиепроцедуры

ИСО/МЭК10164-3—93*Информационнаятехнология.Взаимосвязьоткрытыхсистем.Административноеуправлениеисистемы.Часть3.Функцииадминистративногоуправлениявзаимоотношениями

ИСО/МЭК10164-4—93*Информационнаятехнология.Взаимосвязьоткрытыхсистем.Административноеуправлениеисистемы.Часть4.Функцииуведомлениянештатныхситуациях

ИСО/МЭК11587—96Информационнаятехнология.Взаимосвязьоткрытыхсистем.Применениевконтекстесистемуправленияспроцессамитранзакции

3 Определения

3.1 Определения базовой эталонной модели

В настоящем стандарте использованы следующие термины, определенные в ГОСТРИСО/МЭК7498-1:

- (N)-соединение;**
- (N)-категория;**
- (N)-уровень;**
- (N)-пункт-доступа-к-услуге;**
- открытая система;**
- административное управление системы.**

* Оригиналы и проекты международных стандартов — во ВНИИКИ Госстандарта России.

3.2 Определения наименования и адресации

В настоящем стандарте использован следующий термин, определенный в ГОСТРИСО/МЭК 7498-3:

- (N)-селектор.

3.3 Определения административного управления

В настоящем стандарте использованы следующие термины, определенные в ГОСТРИСО/МЭК7498-4:

- управляемый объект;

- операция (N)-уровня.

3.4 Определения административного управления системы

В настоящем стандарте использованы следующие термины, определенные в ГОСТРИСО/МЭК10040:

- агент;

- родовые определения;

- класс управляемых объектов;

- информация административного управления;

- управляющий;

- протокол административного управления (N)-уровня;

- сообщение;

- тип сообщения;

- операция (административного управления системы);

- (прикладной) протокол административного управления системы.

3.5 Определения модели информации административного управления

В настоящем стандарте использованы следующие термины, определенные в ГОСТРИСО/МЭК10165-1:

- действие;

- фактический класс;

- атрибутивная группа;

- идентификатор атрибута;

- тип атрибутов;

- множество значений атрибута;

- поведение;

- характеристика;

- условный пакет;

- вмещение;

- наследование;

- иерархия наследования;

- управляемый объект начальных значений;

- реализация;

- обязательный пакет;

- кратное наследование;

- связывание имен;

- пакет;

- параметр;

- множество допустимых значений;

- относительно отличающееся имя;

- множество требуемых значений;

- специализация;

- подкласс;

- суперкласс;

- старший объект;

- подчиненный объект.

3.6 Определение УОИУ

В настоящем стандарте используют следующие термины, определенные в ГОСТРИСО/МЭК9595:

а) атрибут;

б) услуги общей информации (административного) управления.

3.7 Определение АСН.1 ИСО/МЭК8824-1:

- а) идентификатор объекта;
- б) тип «последовательность»;
- в) тип «последовательность—из»;
- г) тип множество;
- д) тип «множество—из»;
- е) подтип;
- ж) тип;
- и) имя ссылки на тип;
- к) имя ссылки на значение.

3.8 Дополнительные определения

3.8.1 **определение класса управляемых объектов:** Набор определений атрибутов, операций, общений и поведения, которому присвоено имя класса управляемых объектов в задокументированной и использованной шаблоне класса управляемых объектов одного или нескольких других шаблонов из определенных в настоящем стандарте типов, на которые прямо или косвенно ссылается шаблон класса управляемых объектов. Определение класса управляемых объектов включает себя все элементы определения, наследуемые от суперкласса(ов) этого класса управляемых объектов, и все элементы определения, образующие спецификацию(ии) суперкласса(ов).

3.8.2 **шаблон:** Стандартный формат для документации определения элемента информации административного управления.

3.8.3 **класс объектов справочника:** Класс объектов, определенный в ИСО/МЭК9594-2.

4 Сокращения

В настоящем стандарте использованы следующие сокращения:

- АСН.1 — абстрактная синтаксическая нотация версии 1;
- БДУ — блок данных услуги;
- ВОС — взаимосвязь открытых систем;
- ЗСУО — заявка о соответствии управляемому объекту;
- КУ — качество услуги;
- МФО — методы формального определения;
- ООИ — относительное отличающее имя;
- ПБД — протокольный блок данных;
- ПДУ — пункт доступа к услуге;
- ПК — подкомитет;
- ПОИУ — протокол общей информации (административного) управления;
- РГ — рабочая группа;
- РОУО — руководство по определению управляемых объектов;
- СИУ — структура информации (административного) управления;
- СТК — совместный технический комитет;
- УО — управляемый объект;
- УОИУ — услуги общей информации (административного) управления;
- УОНЗ — управляемый объект начальных значений;
- (N)-ПДУ — (N)-пункт-доступа-к-услуге;

5 Соглашения

В настоящем стандарте шрифтом (полужирным курсивом) выделен текст, в котором использована нотация АСН.1 или определенная в разделе 8.

6 Общие вопросы

6.1 Целостность взаимосвязей

При определении классов управляемых объектов важно рассмотреть ситуации, когда имеются требования согласованности, которые должны применяться как к экземплярам этих классов, например

ситуации, когда поведение управляемого объекта ограничено правилами, зависящими не только от состояния данного объекта, но и от состояний других управляемых объектов в системе. Любые такие ограничения должны быть выражены как поведение, связанное с рассматриваемыми классами управляемых объектов.

Частным случаем, в котором определения, связанные с экземпляром управляемого объекта, должны явным образом устанавливать правила согласования, является операция удаления *elete*. Для этой операции такие правила согласования задаются в связывании (иях) имен, ассоциированном (ых) классе муправляемых объектов. Результат операции *elete* должен быть определен таким образом, чтобы было ясно, при каких обстоятельствах удаление допустимо и какова последовательность удаления. В частности, связывание имен должно устанавливать, допустимо ли удаление экземпляра класса, когда еще существуют содержащиеся в нем управляемые объекты, и какие правила применяются в случаях, когда между удаляемыми и прочими управляемыми объектами есть другие (не относящиеся к вложению) взаимосвязи, как те, которые могут существовать вследствие наличия атрибутов взаимосвязи (см. ИСО/МЭК 10164-3). Применяемые для удаления правила согласованности должны быть такими, чтобы операция удаления не могла привести к несогласованным взаимосвязям. Хотя эти правила согласованности определяются как часть связывания имен, правила, которые применяются для удаления данного управляемого объекта, устанавливаются в момент реализации этого управляемого объекта.

6.2 Наследуемые характеристики

Процесс наследования приводит к включению всех характеристик суперкласса (ов) класса управляемых объектов в определение этого класса. Это правило применяется рекурсивно, завершаясь на вершине иерархии наследования, называемой «высшим классом». Следовательно, данное определение класса управляемых объектов включает всебя все характеристики, которые являются частью определения высшего класса, плюс все характеристики, добавленные в процессе определения тех подклассов высшего класса, которые образуют часть иерархии наследования этого класса управляемых объектов.

6.3 Факультативность

В общем случае включение факультативных возможностей в определение классов управляемых объектов является нежелательным, так как по мере роста числа таких возможностей взаимодействие становится более трудным. Как установлено в ГОСТ ИСО/МЭК 10165-1, определение класса управляемых объектов может содержать условные пакеты, которые присутствуют в экземпляре этого класса, если выполнены заданные условия. Подразумевается, что условия, применяемые для этих пакетов, должны относиться к стандартным свойствам ресурса, который представляет класс управляемых объектов, или к факультативным функциям управления, поддерживаемым управляемой системой.

6.4 Регистрация

Процесс определения классов управляемых объектов требует присвоения глобально однозначных идентификаторов—идентификаторов объектов—различным составляющим класса управляемых объектов, таким как имя класса управляемых объектов, типы атрибутов и пр. Значения этих идентификаторов используются в протоколах административного управления для однозначной идентификации различных сторон управляемых объектов и связанных с ними атрибутов, операций и сообщений. Следовательно, для разработки определения класса управляемых объектов предварительно необходимо, чтобы органы по стандартизации или другие организации идентифицировали или установили подходящий метод регистрации, позволяющий создавать значения идентификаторов объектов. ИСО/МЭК 8824-1 устанавливает структуру идентификатора объекта и значения начальных дуг; дальнейшая информация об установлении методов и полномочных порегистрации приведена в ИСО/МЭК 9834-1.

После того как элементу информации административного управления было присвоено значение идентификатора объекта требуется, чтобы никакой пересмотр определения этого элемента не изменял семантику информации. На практике это означает, что редакционные изменения зарегистрированных определений информации административного управления допускаются, но определения не должны изменяться так, что это будет видно в протоколе.

Все значения идентификаторов объектов, зарегистрированные в международных стандартах по административному управлению системы, размещаются под дугой.

{joint-iso-ccittms(9)}

Распределение дуг ниже {joint-iso-ccittms(9)} определяется настоящим стандартом. Ниже {joint-iso-ccittms(9)} дуги распределяются на основе стандартов по административному управлению системы так, как показано в таблице 1.

ГОСТРИСО/МЭК10165-4—2001

Таблица 1 — Распределение дуг ниже

{joint-iso-ccittms(9)}

Дуга	Стандарт
smo(0)	Основные положения административного управления системы, ГОСТ Р ИСО/МЭК 10040
cmip(1)	Протокол общей информации административного управления, ИСО/МЭК 9596-1
function(2)	Функции административного управления системы, ГОСТ Р ИСО/МЭК 10164-1 и последующие части этого стандарта
smi(3)	Структура информации административного управления, ГОСТ Р ИСО/МЭК 10165-1 и последующие части этого стандарта
applicationContext(4)	Прикладные контексты, ИСО/МЭК 11587

Распределение дуг ниже этого уровня определено в 6.4.1—6.4.5. Дуги, которые потребуются для последующих стандартов по административному управлению системы, будут вводиться по мере необходимости в виде дополнения к настоящему стандарту.

Примечание— Схема распределения значений идентификаторов объектов, описанная в настоящем подразделе, применяется только для значений идентификаторов объектов в стандартах по административному управлению системы, разработанных совместно РГ4ПК21СТК1 ИСО/МЭК и МСЭ-Т. Если другим органом или организацией по стандартизации необходимо в ходе разработки стандартов по административному управлению системы распределять значения идентификаторов объектов, они должны установить свои собственные схемы распределения значений идентификаторов объектов, соответствующие полномочиям по регистрации. Структура, принятая при разработке таких стандартов, может служить в качестве примера того, как устанавливается подходящая схема распределения значений, но окончательный выбор схемы отвечает соответствующая организация. Для обеспечения читаемости значений идентификаторов объектов рекомендуется использовать именную и числовую формы представления значений идентификаторов объектов, как определено в ИСО/МЭК 8824-1.

6.4.1 Распределение идентификаторов объектов для основных положений административного управления системы

Примечание— Выделение этих дуг определяется основными положениями административного управления системы. Здесь они приводятся только в справочных целях.

Ниже **{joint-iso-ccittms(9)smo(0)}** выделены дуги для регистрации идентификаторов прикладных контекстов, абстрактных синтаксисов и модулей АСН.1, приведенные в таблице 2.

Таблица 2 — Распределение дуг ниже

{joint-iso-ccittms(9)smo(0)}

Дуга	Назначение
applicationContext(0)	Распределение идентификаторов прикладных контекстов
negotiationAbstractSyntax(1)	Распределение идентификаторов версий договорных абстрактных синтаксисов
asn1Modules(2)	Распределение идентификаторов модулей АСН.1

Ниже **{joint-iso-ccittms(9)smo(0)applicationContext(0)}**, как установлено в ГОСТ Р ИСО/МЭК 10040, выделены дуги для регистрации идентификаторов конкретных прикладных контекстов, приведенные в таблице 3.

Таблица 3 — Распределение дуг ниже

{joint-iso-ccittms(9)smo(0)applicationContext(0)}

Дуга	Назначение
systems-management(2)	Идентификация прикладных контекстов административного управления системы

Ниже **{joint-iso-ccittms(9)smo(0)negotiationAbstractSyntax(1)}**, как установлено в ГОСТРИСО/МЭК10040, выделены дуги для регистрации конкретных версий договорных абстрактных синтаксисов, приведенные в таблице 4.

Таблица 4 — Распределение дуг ниже

{joint-iso-ccittms(9)smo(0)negotiationAbstractSyntax(1)}

Дуга	Назначение
version(1)	Идентифицирует версию 1 договорного абстрактного синтаксиса

Ниже **{joint-iso-ccittms(9)smo(0)asn1Modules(2)}**, как установлено в ГОСТРИСО/МЭК10040, выделены дуги для регистрации идентификаторов конкретных модулей АСН. 1, приведенные в таблице 5.

Таблица 5 — Распределение дуг ниже

{joint-iso-ccittms(9)smo(0)asn1Modules(2)}

Дуга	Назначение
negotiationefinitions(0)	Распределение идентификаторов версий для модулей АСН. 1, которые содержат определения, связанные с договорным абстрактным синтаксисом

Ниже **{joint-iso-ccittms(9)smo(0)asn1Modules(2)negotiationefinitions(0)}**, как установлено в ГОСТРИСО/МЭК10040, выделены дуги для регистрации конкретных версий модулей АСН. 1, приведенные в таблице 6.

Таблица 6 — Распределение дуг ниже

{joint-iso-ccittms(9)smo(0)asn1Modules(2)negotiationefinitions(0)}

Дуга	Назначение
version1(1)	Идентифицирует версию 1 модуля АСН. 1, который содержит определения, связанные с договорным абстрактным синтаксисом

6.4.2 Распределение идентификаторов объектов для ПОИУ

Примечание— Выделение этих дуг определяется ПОИУ. Здесь они приводятся только в справочных целях. Версия 1 ПОИУ устарела и заменена версией 2. Версия 1 была определена ИСО/МЭК 9596-1 и не имела аналогичной рекомендации МККГТ.

Ниже **{joint-iso-ccittms(9)cmip(1)}** выделены дуги для каждой версии ПОИУ так, как описано в 6.4.2.1 и 6.4.2.2.

6.4.2.1 ПОИУ версии 1

Ниже **{joint-iso-ccittms(9)cmip(1)}** выделены дуги для версии 1 ПОИУ так, как показано в таблице 7.

Таблица 7 — Распределение дуг ниже

{joint-iso-ccittms(9)cmip(1)} для ПОИУ версии 1

Дуга	Назначение
version1(1)	Распределение идентификаторов объектов для ПОИУ версии 1

Ниже **{joint-iso-ccittms(9)cmip(1)version1(1)}** для целей, описанных в ИСО/МЭК 9596-1, выделены дуги так, как показано в таблице 8.

Таблица 8 — Распределение дуг ниже

{joint-iso-ccittms(9)cmip(1)version1(1)}

Дуга
aAssociateUserInfo(1)
aAbortUserInfo(2)
protocol(3)
abstractSyntax(4)

6.4.2.2 ПОИУ версии 2

Ниже **{joint-iso-ccittms(9)cmip(1)}** выделены дуги для версии 2 ПОИУ так, как показано в таблице 9.

Таблица 9 — Распределение дуг ниже

{joint-iso-ccittms(9)cmip(1)} для ПОИУ версии 2

Дуга	Назначение
modules(0)	Распределение идентификаторов объектов для модулей АСН.1 ПОИУ
cmip-pci(1)	Распределение идентификаторов объектов для управляющей информации ПОИУ

Ниже **{joint-iso-ccittms(9)cmip(1)modules(0)}** для целей, описанных в ИСО/МЭК 9596-1, выделены дуги так, как показано в таблице 10.

Таблица 10 — Распределение дуг ниже

{joint-iso-ccittms(9)cmip(1)modules(0)}

Дуга
aAssociateUserInfo(1)
aAbortUserInfo(2)
protocol(3)

Ниже **{joint-iso-ccittms(9)cmip(1)cmip-pci(1)}** для целей, описанных в ИСО/МЭК 9596-1, выделены дуги так, как показано в таблице 11.

Таблица 11 — Распределение дуг ниже

{joint-iso-ccittms(9)cmip(1)cmip-pci(1)}

Дуга
reserved1(1)
reserved2(2)
reserved3(3)
abstractSyntax(4)

6.4.3 Распределение идентификаторов объектов для стандартов функций

Ниже **{joint-iso-ccittms(9)function(2)}** выделены дуги для идентификации стандартов функций так, как показано в таблице 12.

Таблица 12 — Распределение дуг ниже

{joint-iso-ccittms(9)function(2)}

Дуга	Стандарт
partX(X)	(ГОСТР) ИСО/МЭК 10164-Х, где Х — номер части

Дуги ниже **{joint-iso-ccittms(9)function(2)partX(X)}** показаны в таблице 13.

Таблица 13 — Распределение дуг ниже

{joint-iso-ccittms(9)function(2)partX(X)}

Дуга	Назначение
standardSpecificExtension(0)	Специфические для стандарта расширения схемы распределения
functionalUnitPackage(1)	Распределение идентификаторов пакетов функциональных блоков
asn1Modules(2)	Распределение идентификаторов модулей АСН.1
managedObjectClass(3)	Распределение идентификаторов классов управляемых объектов
package(4)	Распределение идентификаторов пакетов
parameter(5)	Распределение идентификаторов параметров
nameBinding(6)	Распределение идентификаторов связываний имен
attribute(7)	Распределение идентификаторов атрибутов
attributeGroup(8)	Распределение идентификаторов атрибутивных групп
action(9)	Распределение типов действий
notification(10)	Распределение типов сообщений
relationshipClass(11)	Распределение идентификаторов классов управляемых взаимосвязей
relationshipMapping(12)	Распределение идентификаторов отображений взаимосвязей
relationshipRole(13)	Распределение идентификаторов ролей взаимосвязей

В любом стандарте функций могут быть выделены дополнительные дуги ниже этого уровня (например для распределения идентификаторов конкретных атрибутов).

6.4.4 Распределение идентификаторов объектов для стандартов СИУ

Ниже **{joint-iso-ccittms(9)smi(3)}** выделены дуги для идентификации стандартов СИУ так, как показано в таблице 14.

Таблица 14 — Распределение дуг ниже

{joint-iso-ccittms(9)smi(3)}

Дуга	Стандарт
partX(X)	(ГОСТР) ИСО/МЭК 10165-Х, где Х — номер части

Дуги ниже **{joint-iso-ccittms(9)smi(3)partX(X)}** показаны в таблице 15.

В любом стандарте функций могут быть выделены дополнительные дуги ниже этого уровня (например для распределения идентификаторов конкретных атрибутов).

Таблица 15— Распределение дуг ниже

{joint-iso-ccittms(9)smi(3)partX(X)}

Дуга	Назначение
standardSpecificExtension(0)	Специфические для стандарта расширения схемы распределения
asn1Modules(2)	Распределение идентификаторов модулей АСН.1
managedObjectClass(3)	Распределение идентификаторов классов управляемых объектов
package(4)	Распределение идентификаторов пакетов
parameter(5)	Распределение идентификаторов параметров
nameBinding(6)	Распределение идентификаторов связываний имен
attribute(7)	Распределение идентификаторов атрибутов
attributeGroup(8)	Распределение идентификаторов атрибутивных групп
action(9)	Распределение типов действий
notification(10)	Распределение типов сообщений
relationshipClass(11)	Распределение идентификаторов классов управляемых взаимосвязей
relationshipMapping(12)	Распределение идентификаторов отображений взаимосвязей
relationshipRole(13)	Распределение идентификаторов ролей взаимосвязей

6.4.5 Идентификатор объекта для фактического класса
Значение идентификатора объекта

{joint-iso-ccittms(9)smi(3)part4(4)managedObjectClass(3)actualClass(42)}

присвоено настоящему стандарту для выражения семантики «фактический класс», определенной в ГОСТРИСО/МЭК10165-1. Когда это значение использовано для спецификации базового класса управляемых объектов в запросе операции УОИУ, оно указывает, что получатель операции административного управления системой должен ответить как член своего фактического класса. Управляемый объект идентифицирует свой фактический класс (см. 7.4.3) с помощью значения своего атрибута «класс управляемого объекта».

6.5 Соответствие

Общие требования соответствия, относящиеся к стандарту информации административного управления, установлены в ГОСТРИСО/МЭК10040.

6.6 Сложность определений управляемых объектов

В процессе моделирования должна быть минимизирована сложность определений управляемых объектов. В любом случае операции административного управления не должны быть более сложными, чем соответствующие свойства участвующей в операции среды ВОС.

6.7 Создание и удаление управляемого объекта

Создание и удаление экземпляров управляемых объектов может происходить следующими методами:

- управляемые объекты могут быть созданы и удалены в результате взаимодействий протокола административного управления. Для этих целей определены операции создания и удаления соответствующей семантикой;

- управляемые объекты могут быть созданы и удалены в результате операций ресурса, к которому они относятся, обычно — протокольного автомата. В этом случае операции создания и удаления не могут быть определены. Примером является представление соединений для целей административного управления;

- управляемые объекты могут быть созданы и удалены другими способами. В этом случае не определены операции создания и удаления. Примером является управляемый объект, который всегда автоматически создается при инициализации части оборудования и который не может быть удален помощью административного управления.

Выбор одного из трех перечисленных методов создания управляемого объекта может отличаться от выбора метода удаления.

В одних случаях может существовать единственный метод, с помощью которого управляемый объект конкретного класса может быть создан и удален. В других случаях может оказаться возможным создание и удаление управляемых объектов конкретного класса разными методами.

6.7.1 Управляемые объекты начальных значений

При создании управляемого объекта может оказаться желательной возможность присвоить значения по умолчанию, которые сами подвержены изменению в результате операций административного управления. Это может быть достигнуто путем спецификации управляемого объекта начальных значений (УОНЗ), атрибуты которого могут изменяться операциями административного управления, который может обеспечивать значения по умолчанию для соответствующих атрибутов создаваемых экземпляров других классов управляемых объектов.

При создании нового управляемого объекта с использованием УОНЗ, значения атрибутов в УОНЗ используются как значения начальных значений соответствующих атрибутов в новом управляемом объекте. Определение класса управляемых объектов может устанавливать, как выбирается УОНЗ. Спецификация УОНЗ должна определять обстоятельства, при которых он предоставляет начальные значения, как он предоставляет эти начальные значения и как атрибутам применимы предоставляемые им начальные значения.

Когда для изменения атрибутов УОНЗ используются операции административного управления, атрибуты созданных ранее с использованием этого УОНЗ управляемых объектов не изменяются. Аналогично операции административного управления, осуществляемые над атрибутами управляемых объектов, созданных с использованием УОНЗ, не влияют на атрибуты УОНЗ.

6.7.2 Источники начальных значений атрибутов

Начальные значения атрибутов управляемых объектов, используемые во время создания, получаются из нескольких источников, как определено в ГОСТРИСО/МЭК10165-1. Когда атрибут представляет конкретное значение, которое должно быть согласованным с ниже лежащим ресурсом, это значение является обязательным.

7 Общие принципы определения управляемых объектов

Описанные ниже общие принципы являются руководством для авторов определений управляемых объектов и должны способствовать согласованности между этими определениями; поэтому авторам определений управляемых объектов рекомендуется по мере возможности следовать данному руководству.

7.1 Общность

Авторы определений управляемых объектов должны стараться идентифицировать и использовать в качестве основы:

- общие классы управляемых объектов, определенные в международных стандартах;
- общие классы управляемых объектов и другие свойства, определенные в ГОСТРИСО/МЭК

10165-2.

Авторы определений управляемых объектов должны так же стараться рассматривать повторно использовать определения, разработанные в других рабочих группах, для увеличения общности определений. Эта цель может быть достигнута путем разработки моделей управления, являющихся общими для нескольких групп определений управляемых объектов.

7.2 Задачи управления

Определения классов управляемых объектов и их компонентов должны полностью удовлетворять требованиям, относящимся к конкретным целям административного управления. Такие требования, вероятно, включают в себя административное управление парными аспектами протокола операций уровня или подуровня и различные проблемы на границе услуги, неговоренные специально поставщиком ниже лежащих услуг (например качество ниже лежащей услуги может не соответствовать приемлемому уровню). В ходе разработки определений классов управляемых объектов важно сохранять техническую обоснованность для каждой цели административного управления. Для

объяснения того, как каждый компонент определения информации административного управления (например классы управляемых объектов, атрибуты, операции, сообщения и пр.) соотносится с этой технической обоснованностью, следует использовать комментарии.

Вопросы, существенные для административного управления, должны быть зафиксированы в управляемых объектах, представляющих ресурсы, к которым относятся эти вопросы, т. е., если должен быть определен управляемый объект, представляющий конкретный ресурс (например соединение), то информация, относящаяся к этому ресурсу, должна быть отражена в соответствующем(их) управляемом(ых) объекте(ах) и нигде более.

7.3 Структурирование

Для представления структуры управляемых объектов имеется ряд способов, позволяющих отразить группирование данных или функциональных возможностей. Каждая из этих способов имеет свои преимущества и недостатки; выбор наиболее подходящих способов для конкретной спецификации зависит от ряда описанных ниже критериев.

Способы структурирования, описанные в ГОСТРИСО/МЭК10165-1, включают всебя:

- атрибутивные группы;
- подклассы (специализацию);
- кратное наследование;
- вмещаемые управляемые объекты;
- пакеты.

Могут быть определены группы атрибутов, операций и сообщений, которые могут присутствовать или отсутствовать в зависимости от стандартизованных условий, таких как выбор конкретных операций в базовом стандарте. Такие группы функциональных возможностей присутствуют или отсутствуют как единое целое. Группы функциональных возможностей могут возникать вследствие факультативного выбора для ресурса из стандарта сля (например обеспечение транспортных услуг класса 4), приводящего к дополнительным требованиям административного управления и дополнительным возможностям, или вследствие обеспечения определенных функций административного управления (например учета). Эти группы функциональных возможностей определяются помощью условных пакетов, предоставляемых шаблоном класса управляемых объектов.

Одним из важных критериев выбора способа структурирования является статическое или динамическое присутствие группы. Если присутствие группы фиксировано на момент спецификации, то подходящим способом может оказаться использование атрибутивных групп, подклассов, кратного наследования или вмещаемых управляемых объектов. Если присутствие фиксировано на момент реализации или установки, то может оказаться подходящим использование вмещаемых управляемых объектов или условных пакетов. Если присутствие группы может изменяться за время жизни вмещающих (или инкапсулирующих) управляемых объектов, то может оказаться подходящим использование вмещаемых управляемых объектов, которые динамически создаются и уничтожаются.

Другим критерием является наличие нескольких экземпляров группы в управляемом объекте. В этом случае подходит использование вмещаемых управляемых объектов; в противном случае может использоваться любой из пяти перечисленных методов.

7.4 Управляемые объекты

7.4.1 Реализация суперклассов

Для обеспечения общей основы, на которой специализируются подклассы, могут быть определены классы управляемых объектов, которые ни когда не реализуются. Например может быть определен родовой класс управляемых объектов «виртуальная цепь», подклассами которого могут быть постоянные и переключаемые виртуальные цепи.

В некоторых случаях, в частности, когда подклассы определяются для пересмотра стандарта, могут существовать суперклассы, экземпляры которых могут быть реализованы.

7.4.2 Неограниченные суперклассы

Правила наследования ограничивают способы, которыми могут быть изменены множества допустимых и требуемых значений атрибутов класса управляемых объектов при определении подкласса этого класса. Точно также правила ограничивают возможности добавления параметров к действиям сообщениям. Эти ограничения гарантируют совместимость подкласса с суперклассом.

По этой причине при определении класса управляемых объектов, который, как ожидается, может быть суперклассом последующих классов управляемых объектов, полезно обеспечить подобногородарасширения. Хотя не все расширения можно предвидеть и обеспечить, следующие приемы оставляют широкие возможности для расширений при сохранении совместимости:

- синтаксис (тип) каждого атрибута следует определять таким образом, чтобы включить в него все значения, которые разумно могут быть согласованы с семантикой атрибута, даже если некоторые из этих значений не являются немедленно необходимыми или желательными;

- следует обеспечивать возможности расширения в каждом определении действия или сообщения;

- следует определять «неограниченный суперкласс», который включает в себя все элементы без каких-либо ограничений, в качестве основы для определения более ограниченных подклассов. Для атрибутов это означает пустое множество требуемых значений и множество допустимых значений, равное синтаксису атрибута;

- следует определять конкретные подклассы этого неограниченного суперкласса, которые устанавливают требуемые ограничения на атрибуты, действия и сообщения.

Авторы определений управляемых объектов могут обеспечивать возможности расширения только для некоторых из атрибутов, действий и сообщений неограниченного суперкласса.

7.4.3 Фактический класс

Определение класса управляемых объектов состоит из шаблона **MANAGE BJECT CLASS** (см. 8.3), зарегистрированного с означением идентификатора объекта для этого класса, набора шаблонов, на которые ссылается данный шаблон, и всех шаблонов, на которые ссылаются шаблоны этого набора.

Управляемый объект идентифицирует свой фактический класс с помощью значения атрибута «класс управляемого объекта», которое является значением идентификатора объекта, использованного для регистрации его шаблона **MANAGE BJECT CLASS**. Каждый управляемый объект:

- обеспечивает все характеристики, определенные для его фактического класса, в соответствии с присутствующими пакетами;

- обеспечивает только те операции, которые определены в его фактическом классе для присутствующих пакетов;

- создает сообщения только тогда, когда поведение, определенное для переключателей этих сообщений в фактическом классе, применяется к присутствующим пакетам.

Отсутствие каких-либо конструкций **POU** для характеристик в определении класса управляемых объектов исключает эти характеристики из определения класса. Подкласс может добавить исключенную конструкцию явно в определении. Каждый подкласс имеет свое собственное зарегистрированное значение идентификатора объекта. Например, если свойство **REPLACE** не задано для однозначного атрибута, то этот атрибут в экземплярах данного класса должен рассматриваться как доступный только для чтения; определение подкласса может расширить исходный класс, добавив конструкцию **REPLACE** для спецификации того, что атрибут может быть заменен в экземплярах подкласса и в экземплярах, совместимых с этим подклассом.

7.5 Атрибуты

7.5.1 Множества значений атрибутов

В некоторых случаях факультативные возможности базового стандарта позволяют изменять множество значений атрибута в соответствии с выбором реализации. Типичным примером является случай, когда базовый стандарт допускает широкий диапазон размеров пакетов, но соответствующая ему реализация может поддерживать более ограниченный диапазон. В такой ситуации определение поведения атрибута должно идентифицировать имеющиеся возможности.

Может оказаться необходимо определить вырожденные значения (**null**) как допустимые значения атрибута или, в случае атрибутов **UON3**, определить значения атрибута с присвоенной им конкретной семантикой, такой как «создать управляемый объект с означением **null** для соответствующего атрибута» или «игнорировать этот атрибут как источник начального значения». Методы определения таких значений включают в себя определение абстрактного синтаксиса выбора рочного типа, когда один выбор определяет нормальное множество значений атрибута, а другой — значения с присвоенной конкретной семантикой.

Определением множества допустимых значений атрибута может быть получено разными способами, включая:

- статическое определение множества значений атрибута, являющееся частью определения класса управляемых объектов;

- определение второго атрибута, значение которого указывает множество значений, которые может принимать рассматриваемый атрибут.

Первый из этих методов минимизирует число определений атрибутов, связанных с классом управляемых объектов; однако если требуется несколько вариантов атрибута, то второй метод позволяет избежать определения нескольких подклассов для обработки каждого возможного варианта множеств значений.

7.5.2 Типы атрибутов

Структурированные атрибуты, в определении синтаксиса которых в качестве базового использован тип «последовательность», «последовательность-из» или «множество», должны использоваться только тогда, когда не требуется изменений отдельных элементов атрибута, т. е. эти типы АСН. I соответствуют типам атрибутов с единственным значением. Когда необходимо обращаться к нескольким атрибутам вместе, обеспечивая при этом возможность работать с каждым из них по отдельности, могут быть определены атрибутивные группы и, при необходимости, могут быть использованы определения действий и поведения для установления любых зависимостей между членами группы.

Примечание—Не подразумевается, что есть особая спецификация поведения для самой атрибутивной группы, которая не применяется к атрибутам, рассматриваемым по отдельности.

7.6 Взаимосвязи значений атрибутов

Значение атрибута может быть ограничено некоторыми функциями других значений атрибутов. Все взаимосвязи такого рода должны быть идентифицированы.

Когда значение атрибута ограничено другими атрибутами в том же самом управляемом объекте, могут существовать требования синхронизации для операции управления, в соответствии с которыми изменения значений одного или нескольких связанных атрибутов, приводящие к недопустимым значениям в связанных атрибутах, вызывают отказ. Если такие требования существуют, то они должны быть задокументированы как часть определения поведения класса управляемых объектов.

Когда значение атрибута ограничено другими атрибутами в разных управляемых объектах (если требования синхронизации также существуют), это должно быть задокументировано в поведении, связанном с классом управляемых объектов. В этом случае, когда все управляемые объекты находятся в одной и той же управляемой системе и один атрибут может изменять одна операция управления, требования реализуются через возможность элементарной межобъектной синхронизации УОИУ.

Общие проблемы синхронизации между несколькими операциями управления, между разными атрибутами в разных управляемых объектах или между несколькими управляемыми системами не могут быть решены с помощью современных протоколов административного управления систем.

7.7 Моделирование ПДУ

Существует общее требование—представлять как часть связанной со слоем структуры управляемого объекта, взаимосвязи между (N)-категориями, (N)-селекторами и (N+1)-категориями. Возможны различные решения, например:

- моделирование взаимосвязи как информации, содержащейся в управляемых объектах (N+1)-уровня;

- моделирование взаимосвязи как информации, содержащейся в управляемых объектах (N)-уровня;

- моделирование взаимосвязи как информации, содержащейся в управляемых объектах, которые не относятся ни к какому уровню, например в управляемых объектах, общих для всех уровней.

Настоящим стандартом рекомендуется второй из перечисленных подходов. В частности, (N)-ПДУ должны быть представлены отдельными управляемыми объектами, которые имеют в качестве атрибута адреса (и другую информацию), вместе с взаимозаменяемыми атрибутами, указывающими на управляемые объекты (N)- и (N+1)-категории, ассоциированные с (N)-ПДУ. Для реализации требования согласованности селекторов, необходимых для недвусмысленной адресации ВОС, рекомендуется, чтобы управляемые объекты (N)-ПДУ содержались в управляемых объектах, соответствующих (N)-категориям, с которыми они связаны.

Примечание—Названное выше требование согласованности состоит в том, что адрес (N)-категории вместе с (N)-селектором должен однозначно идентифицировать (N+1)-категорию (или набор (N+1)-категорий одного и того же типа). Принимая, что это требование равносильно требованию однозначности для присвоения значений (N)-селекторов в контексте данной (N)-категории, обеспечение требования согласованности может быть более просто достигнуто, если информация селектора обеспечивается (N)-, а не (N+1)-категорией.

7.8 Статистика

7.8.1 Согласованность

Авторы определений управляемых объектов должны стараться достичь согласованности с статистикой уровней, принимая принципы ГОСТРИСО/МЭК7498-1; в частности, понятие «регистрируемая информация» относится к информации, рассматриваемой при административном управлении через управляемые объекты, представляющие ресурсы, к которым относится информация.

Кандидатами в характеристики (N)-слоя, статистика которых может регистрироваться, относятся:

- локальные ошибки;
- успешный равноправный обмен;
- взаимные отказы;
- отказы в услуге.

Например, применение определенных выше принципов к соединению, определенному в ГОСТРИСО/МЭК7498-1, приводит к идентификации следующих основных характеристик:

- число установленных соединений (N)-категорий с другими парными категориями (N)-слоя;
- число локальных отказов при установленных соединениях (N)-категорий;
- число отказов при взаимных согласованиях для соединений (N)-категорий;
- число отвергнутых (N-1)-соединений поставщиками ниже лежащих услуг.

Этот набор статистических характеристик обеспечивает согласованный взгляд на то, что происходит на каждом уровне (в случае, ориентированном на соединения) без дублирования счетчиков.

Примечание — Аналогичные модели требуются для ошибок, разрывов соединений и пр.

7.8.2 Счетчики ПБД

Авторы определений управляемых объектов должны специфицировать счетчики ПВД (N)-уровня (и октетов ПБД), а не БДУ (и октетов БДУ).

Примечание — Вероятно требуется подсчитывать только число октетов ПБД на ограниченном числе (N)-уровней.

7.8.3 Перекрытия

Авторы определений управляемых объектов должны стараться достигнуть согласованности и избегать излишнего дублирования или перекрытия статистической информации. Например, обеспечивая подсчет запросов отправки первого ПБД и запросов отправки ответного ПБД, необязательно увеличивать оба счетчика одновременно. Сумма этих счетчиков всегда равна общей отправке ПБД.

7.8.4 Непереустанавливаемые счетчики

Рекомендуются непереустанавливаемые счетчики, так как они допускают многократное наблюдение без необходимости сложных механизмов взаимной блокировки, связанных с координацией переустановки.

7.8.5 Счетчики событий

Должен быть обеспечен подсчет событий административно управляемого ресурса, которые приводят к созданию сообщений, т.к. создание УОИУМ—EVENT—REPORT может быть подавлено опережающими дискриминаторами событий.

7.9 Счетчики

Для административного управления счетчиками должны быть известны модули, т.к. в противном случае управляющий не сможет определить, когда сбрасывается счетчик. Следовательно, имеются, по крайней мере, четыре возможности определения счетчиков:

- счетчики никогда не сбрасываются;
- модули фиксированы как часть определения класса управляемых объектов;
- модули определяются соответствующими атрибутами;
- модули определяются реализацией и специфицированы в ЗСУО.

Примечание — Для классов управляемых объектов, определенных СТК1 ИСО/МЭК для уровней 1—4, принят подход, при котором счетчики никогда не сбрасываются.

7.10 Таймеры

Преимущества могут быть получены при наследовании общей спецификации точности, с которой системы должны сохранять значения атрибутов таймеров, используемых при взаимодействии.

ых административного управления. Взаимосвязи между значениями этих атрибутов фактически операциями таймеров в протоколах документируются в описании поведения.

Примечание — Для классов управляемых объектов, определенных СТК1 ИСО/МЭК для уровней 1—4, с целью обеспечения достаточно большого диапазона без экстраординарной точности для выражения значений таймеров использовано представление сплавующей точкой с мантиссой длиной 16 бит и экспонентой длиной до 16 бит. (Это не подразумевает, что должна использоваться арифметика с плавающей точкой). Системы должны быть в состоянии сохранять значения с этой точностью. Допуская другие ограничения, должно быть принято требование устанавливать атрибут таймера с этой точностью.

7.11 Обновление атрибутов

Авторы определений классов управляемых объектов должны гарантировать, что при определении атрибутов, которые могут обновляться операциями управления и обычными операциями ресурса, определены результаты конкурирующих обновлений. В частности, результатом операции замены значения атрибута может быть потеря, если ресурс обновляет тот же самый атрибут.

7.12 Точность атрибутов

Управляющая система может попытаться установить значение атрибута с большей точностью, чем обеспечивается управляемой системой. Такие высокие точные значения могут быть аппроксимированы ближайшим значением установленной точности.

7.13 Идентификация управляемого объекта

Каждое определение класса управляемых объектов, экземпляр которого могут существовать, должно включать себя, по крайней мере, один атрибут, пригодный для использования в качестве именуемого атрибута управляемого объекта. Подходящий атрибут является обязательным и может быть проверен на равенство; его семантика должна допускать сохранение фиксированного значения на протяжении времени жизни каждого управляемого объекта, использующего атрибут для именованного; его идентификатор значения должен однозначно идентифицировать управляемый объект среди всех других, поименованных тем же самым старшим объектом.

При удалении управляемого объекта значение, присвоенное его именуемому атрибуту, становится доступным для повторного использования целью идентификации управляемых объектов, создаваемых в дальнейшем в том же самом старшем объекте.

Если не обязательно гарантировать, что экземпляр класса управляемых объектов после удаления остается отличным от всех других экземпляров этого класса, то следует определить дополнительный атрибут, входящий в определение класса управляемых объектов, — атрибут однозначной идентификации, — семантика которого обеспечивает однозначную идентификацию во времени. Другие классы управляемых объектов не обязаны содержать атрибут однозначной идентификации.

Атрибут однозначной идентификации должен быть доступен только для чтения и, когда он входит в управляемый объект, должен включаться в сообщения, создаваемые этим объектом.

7.14 Сообщения

7.14.1 Отказ услуги

Не должны создаваться сообщения, относящиеся к отказам ниже лежащей услуги, так как за сообщения о любых причинах такого ненормального завершения должен нести ответственность управляемый объект, представляющий ниже лежащую услугу. Это условие должно предотвратить распространение вверх по уровням ненормального завершения и генерацию ложных сообщений.

7.14.2 Сохранение информации

Сообщения содержат информацию о событии, которая иначе могла бы быть потеряна. Например:

- заголовок поля полученного ПБД, для которого была обнаружена ошибка протокола;
- статистические данные о соединении, которое должно быть завершено;
- время, в течение которого происходит последовательность конкретных событий.

7.15 Использование операций

Определение класса управляемых объектов должно включать в себя соответствующие операции. Для вызовов управляемой системой должны быть определены сообщения. Для вызовов управляющей системой операции специфицируются в соответствии с их непосредственным влиянием на управляемые объекты в управляемой системе следующим образом:

а) если непосредственным результатом является создание экземпляра класса управляемых объектов, то используется операция Create. Операция Create не используется: для сложных действий, которые требуют скоординированного создания нескольких управляемых объектов; когда управляемый объект создается как побочный результат изменения другого управляемого объекта; когда управляемые объекты создаются в результате изменения состояния другого управляемого объекта;

б) если непосредственным результатом является удаление управляемого объекта, то используется операция `delete`;

в) если непосредственным результатом является установление значений атрибутов управляемого объекта равными заданным значениям, то используется операция `Replace—attribute—value`;

г) если непосредственным результатом является установление значений атрибутов управляемого объекта равными значениям по умолчанию (при условии, что такие значения были определены), то используется операция `Replace—with—default—value`;

д) если непосредственным результатом является добавление или удаление членов многозначных атрибутов управляемого объекта, то используется операция `Add—member` или `Remove—member`;

е) если непосредственным результатом является получение от управляемого объекта значений атрибутов, то используется операция `Get—attribute—value`;

ж) во всех других случаях, например когда нет непосредственного результата или непосредственный результат является комбинацией перечисленных выше, или имеется какое-либо влияние на объект в целом, используется операция `Action`. Примерами использования этой операции являются случаи, когда:

1) невозможно определить требуемую операцию над множеством управляемых объектов с использованием области действия и фильтров вместе с операциями `Get—attribute—value`, `Replace—attribute—value`, `Replace—with—default—value`, `Create`, `delete`, `Add—member` или `Remove—member`;

2) требуется качество элементарной операции создания нескольких управляемых объектов;

3) влияние оказывается на несколько объектов без общих атрибутов;

4) запросил ответ операции и содержит информацию, которая не может моделироваться атрибутами управляемых объектов.

Понятия непосредственного и побочного результатов рассмотрены в ГОСТРИСО/МЭК 10165-1.

8 Обозначения для определений управляемых объектов

8.1 Обзор обозначений

Определенные в настоящем разделе шаблоны обеспечивают общий набор обозначений для представления различных аспектов определений классов управляемых объектов в связанных с ними структурных наименованиях. Формальные определения шаблонов содержатся в 8.3—8.11; использованные в этих формальных определениях синтаксические соглашения описаны в 8.2. Эти формальные определения устанавливают конструкции, которые могут или должны содержать каждый шаблон, и порядок, в котором конструкторы должны появляться в шаблоне. Примеры использования этих обозначений приведены в приложении А.

Структура и поведение класса управляемых объектов определяются, в основном, с помощью шаблона класса управляемых объектов. Шаблоны идентифицируют взаимосвязи наследования, которые существуют между определяемыми другими классами управляемых объектов, и пакеты поведения, атрибутов, сообщений и операций, которые включаются в определение класса управляемых объектов. Для повторного использования частей данной спецификации, в спецификациях других классов управляемых объектов определены дополнительные шаблоны, обеспечивающие спецификацию атрибутов (отдельных и в группах), поведения, действий, сообщений, параметров и пакетов. Эти дополнительные шаблоны являются «вызываемыми» другими шаблонами с помощью метода ссылок, определенного в 8.2. Этот метод позволяет ссылаться из любого стандарта на спецификацию, содержащуюся в других стандартах, допуская, таким образом, использование родовых определений для определений классов управляемых объектов. Эти дополнительные шаблоны, при желании, могут быть включены в тело определения.

Наименование класса управляемых объектов определяется с помощью шаблона связывания имен. Этот шаблон идентифицирует именованный класс управляемых объектов и определяет относительное отличающее имя, которое может быть использовано для присвоения имен экземплярам класса в контексте конкретного старшего класса. Этот шаблон обеспечивает спецификацию взаимосвязей, существующих между двумя классами объектов в результате связывания имен.

8.2 Соглашения, использованные в определениях шаблонов

Шаблон начинается с метки-шаблона и имени шаблона `TEMPLATE—NAME`. Шаблон содержит одну или несколько конструкций, каждая из которых имеет имя `CNSTRUCT—NAME` и

может иметь аргумент-конструкции. Аргумент-конструкции может состоять из нескольких элементов для вызова из определения конкретной конструкции. Для каждого использования шаблона декларируется уникальная метка-шаблона, с помощью которой можно ссылаться из других шаблонов на данный экземпляр шаблона, и, если присутствует конструкция **REGISTERED AS**, присваивается значение идентификатора объекта АСН.1, под которым зарегистрирован данный экземпляр использования шаблона. Символ «;» используется в качестве признака конца каждой конструкции (кроме **REGISTERED AS** и **EFINE AS**) и конца шаблона.

Для упрощения структуры шаблонов, например, когда одна и та же синтаксическая структура неоднократно используется в определении шаблона, могут быть введены определения обеспечивающих синтаксисов. Если такие определения нужны, то они вводятся с помощью ключевых слов **supporting productions**

в конце определения шаблона и состоят из продукции вида

метка-определения j -j синтаксическое-определение j

Метка-определения позволяет ссылаться на определение из определения шаблона или из других обеспечивающих синтаксических продукций, а синтаксическое-определение дает раскрытие определения, используя установленные синтаксические соглашения. В случае синтаксического-определения, специфицирующего несколько альтернативных строк, принято, что ссылка на содержащую его синтаксическую продукцию должны вычисляться до единственной строки, выбранной из этих альтернатив.

Определения шаблонов основаны на следующих синтаксических соглашениях:

а) все терминальные символы и ключевые слова, образующие часть определения шаблона, зависят от регистра;

б) когда необходимо для недвусмысленной передачи синтаксиса шаблона, элементы шаблона должны отделяться от соседних одним или несколькими разделителями. Допустимыми разделителями являются пробел, конец строки, пустая строка и комментарий. Один или несколько разделителей должны присутствовать между:

- 1) меткой-шаблона и **TEMPLATE-NAME**;
- 2) **TEMPLATE-NAME** и **CNSTRUCT-NAME**;
- 3) **CNSTRUCT-NAME** и аргументом-конструкции.

Между любой парой элементов в шаблоне может быть вставлено один или несколько разделителей, а когда аргумент-конструкции состоит из нескольких различных элементов — разделители могут быть вставлены между ними. Разделители могут быть вставлены внутри элементов шаблона, если только определение шаблона явно позволяет это сделать;

в) пробелы, пустые строки, комментарии и концы строк имеют смысл только как разделители;

г) комментарий начинается и завершается парой символов или концом той строки, в которой встретилась первая пара. При интерпретации шаблонов, определенных в настоящем стандарте, комментарий эквивалентен пробелу. Комментарий не имеет нормативного значения;

д) символ

;

должен отмечать конец каждой конструкции в шаблоне (кроме **REGISTERED AS** и **EFINE AS**) и конец самого шаблона;

е) для представления идентификаторов объектов должна использоваться нотация, определенная в ИСО/МЭК 8824-1, например продукция

идентификатор-объекта-j **ObjectIdentifierValuej**

представляет продукции для всех определений шаблонов настоящего стандарта, а **ObjectIdentifierValue** указывает на соответствующую нотацию, определенную в ИСО/МЭК 8824-1;

ж) строки, ограниченные парой

[]

выделяют в определении шаблона части, которые могут присутствовать или отсутствовать в конкретных случаях использования шаблона. Если закрывающей скобкой следует звездочка

[]★

то содержимое скобок может появляться нуль или несколько раз. Обстоятельства, при которых данные части определения могут быть опущены или повторены, зависят от определения типа шаблона;

з) строки, ограниченные парой

к j

выделяют в определении шаблона строки, которые должны быть заменены в конкретных случаях использования шаблона. Структура смысла подставляемой строки зависит от ее типа;

и) прописные строки обозначают ключевые слова, которые обязательно должны присутствовать в каждом случае использования шаблона, если они не заключены в квадратные скобки

[]

для указания их факультативности;

к) символ

|

используется как разделитель альтернативных строк синтаксических определений обеспечивающих производций **supporting productions**. Когда обеспечивающая продукция используется для определения альтернативных строк, открывающим разделителем первой альтернативы является `-j`, символ `|` является закрывающим и открывающим символом для последующих альтернатив, а закрывающим разделителем последней альтернативы является первый конец строки после ее открывающего разделителя;

л) метка-шаблона должна быть уникальной в пределах стандарта или документа, в котором она объявлена. В стандарте или документе, состоящем из нескольких частей, которые сопровождаются и распространяются по отдельности, метка-шаблона должна быть уникальна в пределах той части, где она объявлена.

Требование уникальности метки-шаблона не зависит от типа помечаемого шаблона. Например, если метка **labell** использована в документе для экземпляра использования некоторого шаблона, то не допускается помечать меткой **labell** экземпляра использования шаблона того же самого или другого типа.

Когда метка-шаблона объявлена в документе А и указывается из документа В, то ссылка в документе В должна иметь в качестве префикса глобально однозначное имя документа А. В случае документов, названных международно признанными уполномоченными по наименованиям, такими как МККТТ или ИСО/МЭК, должны использоваться в качестве идентификаторов зарегистрированные обозначения документов, такие как «**СCITT Rec. X.722 (1992) | IS / IEC 10165-4: 1992**». Формат этой строки должен быть установлен уполномоченным по наименованиям для рассматриваемого документа. Когда рассматриваемый документ совместно разработан и опубликован МККТТ и ИСО/МЭК, обозначение документа должно содержать оба обозначения, разделенные знаком «|», как показано в приведенном примере. Если глобально однозначное имя не существует, допускается присвоение указываемому документу значения идентификатора объекта и использование этого значения в качестве глобально однозначного имени документа. Определенный выше синтаксис метки-шаблона выглядит следующим образом:

[идентификатор-документа:]к строка-меткиj

идентификатор-документа —

«<имя-стандарта>|идентификатор-объекта

Строка-метки может включать всебя любое число следующих символов:

1) прописные и строчные алфавитные символы;

2) цифры 0—9;

3) символ

-

4) символ

/

в любом порядке, начиная со строчного алфавитного символа, за исключением того, что пара символов

--

недолжна появляться в строке-метке. Например, следующая метка-шаблона

«**СCITT Rec. X.722 (1992) | IS / IEC 10165-4: 1992**»: **exampleObjectClass**

является глобально однозначной меткой для определения **exampleObjectClass** в приложении А.

Ссылка на метку без префикса идентификатор-документа указывает на метку, объявленную в том же документе, что и ссылка.

м) в любом месте шаблона, где метка-шаблона присутствует как указание на другой шаблон, она может быть заменена полным текстом указанного шаблона (включая метку-шаблона). Это по-

звоняет включать в тело шаблона другие шаблоны, на которые он ссылается (подшаблоны), при сохранении возможности для этих указываемых шаблонов, в свою очередь, ссылаться на подшаблоны. В результате обеспечивающая продукция **supporting production**

метка-шаблона-определение-шаблона

используется для всех экземпляров метки-шаблона и определения-шаблона;

н) когда необходимо сослаться из шаблона на определение значения или типа АСН.1, имя типа или значения АСН.1 имеет в качестве префикса имя модуля АСН.1, содержащего определение этого типа или значения. При этом принимается, что имя модуля относится к модулю АСН.1, находящемуся в том же самом документе, что и шаблон, из которого дается ссылка на тип или значение. Следовательно, обеспечивающие продукция

указание-типа - <имя-модуля.<имя-типа

указание-значения - <имя-модуля.<имя-значения

используются для всех определений шаблонов, которые ссылаются на типы или значения АСН.1, где имя-модуля — имя, присвоенное модулю АСН.1 в документе, содержащем ссылку, а имя-типа и имя-значения — имена, присвоенные определениям типов или значений АСН.1, содержащимся в этом модуле. Если необходимо сослаться на определения типов или значений, содержащиеся в других документах, то это можно сделать с помощью локального модуля АСН.1, который использует утверждение **IMP RT** для импорт соответствующих определений типов или значений. (См. раздел 9.)

о) когда в шаблон необходимо включить текст, он включается в виде строки символов, фактативно начинающейся и заканчивающейся символом разделитель-текста, выбранного из последующих символов:

! " # \$ % ^ & * ' ' ? @ \

Если используется символ разделитель-текста, то один и тот же символ должен использоваться в начале и в конце строки, а если этот разделитель-текст встречается в теле текстовой строки, то он должен быть заменен двойным вхождением символа. Если разделитель-текст не используется, то строка не должна содержать никаких знаков пунктуации, которые являются допустимыми для текстовых строк в этом определении шаблона.

Следовательно, обеспечивающие продукция

выделенная-строка-

разделитель-текста<текстовая-строка разделитель-текста|<текстовая-строка

разделитель-текста - ! | " | \$ | % | ^ | & | * | ' | ' | | ? | @ | \

используются для всех шаблонов, которые допускают выделенные-строки, где текстовая-строка — произвольная последовательность символов; если используется разделитель-текста, то все появления этого символа в текстовой-строке должны быть заменены парой символов разделитель-текста.

За исключением правил, относящихся к использованию разделителей, внутренняя структура текстовой-строки, в частности, использование определенной в настоящем стандарте структуры комментария, не имеет отношения к положениям настоящего стандарта.

8.3 Шаблон класса управляемых объектов

8.3.1 Обзор

Шаблон класса управляемых объектов является основой для формального определения управляемого объекта. Элементы шаблона позволяют разместить класс в подходящем узле дерева наследования, специфицировать различные характеристики класса и определить поведение класса. Ниже определено большинство этих элементов.

8.3.1.1 Наследование

Каждый класс управляемых объектов определяет суперкласс(ы), из которого(ых) он выводится. Характеристики суперкласса(ов) наследуются подклассом; определение подкласса может добавлять характеристики (специализация подкласса), но не может исключать характеристики суперкласса. Все классы являются подклассами высшего класса.

8.3.1.2 Обязательные пакеты

Определение класса управляемых объектов включает в себя пакеты поведения, атрибутов, операций и сообщений, которые обеспечивают полную спецификацию поведения, характеризующего все экземпляры класса.

8.3.1.3 Условные пакеты

Определение класса управляемых объектов включает в себя пакеты поведения, атрибутов, операций и сообщений, которые присутствуют в экземплярах этого класса вследствие заданного условия.

8.3.1.4 Наименование класса

Определение класса управляемых объектов включает в себя имя класса, которое может использоваться в протоколе административного управления для ссылок на класс. Это достигается регистрацией значения идентификатора объекта, присвоенного определению класса управляемых объектов.

8.3.2 Структура шаблона

```

<метка-класса      MANAGE BJECT CLASS
[ ERIVE FROM      <метка-класса
                        [, <метка-класса]*;
]
[CHARACTERIZE BY <метка-пакета
                        [, <метка-пакета]*;
]
[CNITINAL PACKAGES <метка-пакета
                        PRESENTIFопределение-условия
                        [, <метка-пакета
                        PRESENTIFопределение-условия]*;
]
REGISTERE ASидентификатор-объекта;
supporting productions
определение-условия-выделенная-строка

```

8.3.3 Обеспечивающие определения

8.3.3.1 **ERIVE FR M** <метка-класса[, <метка-класса]*

Конструкция **ERIVE FR M** должна присутствовать во всех определениях классов управляемых объектов, кроме высшего. Следовательно, высший является суперклассом для всех классов управляемых объектов, кроме самого себя.

Метка-класса идентифицирует класс управляемых объектов, из которого выводится определяемый класс, т. е. тот класс управляемых объектов, который является одним из непосредственных суперклассов определяемого класса. Так как допустимо кратное наследование, класс управляемых объектов может иметь несколько непосредственных суперклассов.

Процесс наследования (специализации) требует, чтобы все характеристики суперкласса(ов) были включены в определение подкласса.

Характеристики, которые наследуются от суперкласса, не должны повторяться в документации подкласса, если не используется ни один из описанных в настоящей стандартной процедуре расширения изменения определения, наследуемого из суперкласса. Следовательно, конструкция **ERIVE FR M** подразумевает автоматический импорт всех характеристик определения(ий) суперкласса(ов). Эти характеристики могут быть расширены или изменены элементами, определенными в конструкциях **CHARACTERIZE BY** и **CNITINAL PACKAGES**.

Примечание 1 — Перечень всех классов управляемых объектов, характеристики которых наследует определяемый класс, рекомендуется в виде комментария включать в документацию определения класса управляемых объектов.

Когда в результате наследования несколько раз импортируется определение одного и того же элемента (что может произойти, например, если два определения суперклассов содержат один и тот же атрибут), принимают, что подкласс содержит единственную копию рассматриваемого определения.

Сточки зрения разрешения конфликтов, которые могут существовать между элементами, определенными в пакетах, и условными пакетами, наследуемыми или включаемыми в определение класса управляемых объектов при специализации, все пакеты, которые должны быть включены в конкретный класс управляемых объектов, рассматриваются как идентичные. Каждый пакет определяет несколько элементов, которые трактуются следующим образом:

а) **BEHAVIUR**. Пакеты, включаемые в подкласс, расширяют наследуемое поведение. Поведение класса управляемых объектов должно быть выражено таким образом, чтобы учитывать возможное присутствие или отсутствие условных пакетов. Когда поведение расширяется, то установленные или подразумеваемые предусловия могут быть только ослаблены (обязательные предусловия должны оставаться теми же или их число может быть уменьшено), установленные или подразумеваемые

емые постусловия могут быть только усилены (должны удовлетворяться те же постусловия, могут удовлетворяться дополнительные), а установленные или подразумеваемые инварианты остаются неизменными, но могут быть добавлены новые (см. ГОСТРИСО/МЭК10165-1, 5.2.2.6);

Примечание 2 — При некоторых обстоятельствах могут быть определены подклассы, которые не требуют определения дополнительного поведения сверх того, которое наследуется от суперкласса(ов);

б) **ATTRIBUTES**. В пакетах, включаемых во определение подкласса, могут быть специфицированы пакеты. Когда конструкция **ATTRIBUTES** в пакете идентифицирует атрибут, который неоднократно определен в классе управляемых объектов, применяются следующие правила:

1) в реализованный управляемый объект должен быть включен единственный атрибут этого типа,

2) результирующий список-свойств логическое или (**R**) включенных в подкласс и наследуемых списков-свойств, за исключением свойств **PERMITTED VALUES**, для которого реализуемое множество допустимых значений является пересечением всех спецификаций допустимых значений для этого типа атрибута, и **REQUIRE VALUES**, для которого реализуемое множество обязательных значений является пересечением реализуемого множества допустимых значений с объединением всех спецификаций обязательных значений для этого типа атрибута. Если для свойства атрибута **EFAULT VALUE** и **INITIAL VALUE** в совокупности определений заданы противоречивые значения, то включаемый в подкласс пакет должен разрешать этот конфликт,

3) параметры, связанные с данным атрибутом, являются объединением всех параметров, связанных с шаблоном атрибута, и всех параметров, связанных с атрибутом во всех тех пакетах, которые реализуются.

Если класс управляемых объектов предназначен для реализации, то должен быть определен, по крайней мере, один атрибут как часть определения класса, так как необходимо идентифицировать атрибут, который может быть использован для имен экземпляров управляемых объектов,

Примечание 3 — Атрибуты, используемые для наименования, могут быть выбраны из числа любых атрибутов, которые являются частью определения класса. В их число входят все атрибуты, наследуемые от суперклассов и добавленные к классу в результате специализации;

в) **ATTRIBUTE GRUPS**. Для расширяемой атрибутивной группы множество ее членов в экземпляре подкласса является объединением всех атрибутов, определенных в шаблоне атрибутивной группы и добавленных к этой группе в суперклассе(ах) или в подклассе;

г) **ACTINS**. Во определение подкласса могут быть включены действия; это могут быть действия в дополнение к наследуемым от суперклассов, или они могут включать дополнительные параметры для наследуемых действий. Множество параметров, связанных с данным действием, является объединением всех параметров, связанных с шаблоном действия и с действием во всех тех пакетах, которые реализуются;

д) **NTIFICATIONS**. Во определение подкласса могут быть включены сообщения; это могут быть сообщения в дополнение к наследуемым от суперклассов или они могут включать дополнительные параметры для наследуемых сообщений. Множество параметров, связанных с данным сообщением, является объединением всех параметров, связанных с шаблоном сообщения и с сообщением во всех тех пакетах, которые реализуются.

Если пакет включен во определение класса управляемых объектов несколько раз, путем наследования и (или) неоднократным включением в шаблон класса управляемых объектов, то результирующее определение-условия, связанное с пакетом, является логическим ИЛИ всех определений-условий в совокупном множестве определений. Для этой цели пакеты, входящие в конструкции **CHARACTERIZE BY** (обязательные пакеты), рассматриваются как входящие в конструкцию **CNITIAL PACKAGES** с определением условия **PRESENTIF[TRUE]**.

Характеристики в (обязательном или условном) пакете могут зависеть от характеристик других условных пакетов, если только связанные с этими пакетами условия гарантируют, что требуемые характеристики будут присутствовать во всех управляемых объектах, в которых присутствует первый пакет.

8.3.3.2 **CHARACTERIZE BY**<метка-пакета [, <метка-пакета]*

Данная конструкция, если она присутствует, позволяет включить во определение класса управляемых объектов один или несколько обязательных пакетов поведения, атрибутов, операций и

сообщений в дополнение к тем, которые являются частью определения в результате конструкции **ERIVE FR M**. Метка-пакета идентифицирует определение того пакета, который должен быть включен. Спецификация метки пакета, который включен в определение класса управляемых объектов как условный пакет, делает этот пакет обязательным для данного класса управляемых объектов во всех его подклассах.

8.3.3.3 **C N I T I N A L P A C K A G E S** <метка-пакета **PRESENTIF** определение-условия[, <метка-пакета **PRESENTIF** определение-условия]*

Данная конструкция присутствует, если в класс должны быть включены один или несколько условных пакетов. Метка пакета идентифицирует определение того пакета, который используется. Определение-условия является описанием условия, которое, если оно истинное, требует, чтобы пакет был включен в экземпляр класса. Условие должно удовлетворять требованиям к условным пакетам, установленным в ГОСТРИСО/МЭК10165-1. Например,

C N I T I N A L P A C K A G E S class-4-attributes PRESENTIF
соответствующая категория протокола поддерживает операцию класса4, определенную в ИСО/МЭКXXXX' ;

образует допустимую декларацию пакета при условии, что операция класса4, определенная в стандарте ИСО/МЭКXXXX, является допустимой факультативной характеристикой ресурса.

Когда имеются специфичные условия, которые препятствуют реализации условного пакета, они должны быть установлены в шаблоне **BEHAVIUR**. Используемый для этого шаблон **BEHAVIUR** может быть в самом условном пакете или в обязательном пакете класса. Если такие спецификации присутствуют в текстовых определениях поведения, то рекомендуется, чтобы абзац, содержащий эти спецификации, вводился конструкцией "<метка-пакета **PRESENT NLY IF**: ".

8.3.3.4 **REGISTERE AS** идентификатор-объекта

Значение идентификатора-объекта обеспечивает глобально однозначный идентификатор определения класса управляемых объектов. Это значение используется протоколом административного управления для идентификации класса управляемых объектов.

8.4 Шаблон пакета

8.4.1 Обзор

Данный шаблон позволяет определить пакет, состоящий из комбинации определений поведения, атрибутов, атрибутивных групп, операций, сообщений и параметров, для последующей вставки шаблона класса управляемых объектов в конструкции **CHARACTERIZE BY** или **C N I T I N A L P A C K A G E S**. Ниже описаны основные элементы определения.

8.4.1.1 Поведение

Определение пакета обеспечивает полную спецификацию поведения, входящего в пакет. Она включает себя:

- влияние операций на управляемый объект обстоятельства, при которых создаются сообщения;
- ограничения, которые накладываются на операции для удовлетворения правил согласованности и, в частности, правилам, по которым может осуществляться создание и удаление управляемых объектов и последовательности этих операций;
- спецификацию того, как экземпляры класса управляемых объектов взаимодействуют с другими управляемыми объектами того же или других классов;
- идентификацию любых атрибутов, которые соотносятся с информацией в сообщениях. Это включает себя идентификацию любых отображений в конкретные поля создаваемых сообщений или самосоздание сообщений;
- спецификацию критериев выбора УОНЗ, если они есть;
- полное определение любых других аспектов поведения класса управляемых объектов.

8.4.1.2 Включаемые атрибуты

Должно быть определено множество атрибутов, которые содержат пакет.

8.4.1.3 Операции и сообщения

Определение пакета должно специфицировать, какие могут создаваться экземпляры сообщений класса, использующего этот пакет, какие могут осуществляться экземпляры операций класса и, в случае операций, относящихся к атрибутам, над какими атрибутами могут осуществляться операции. Определение пакета должно так же специфицировать любые дополнительные параметры

тех сообщений и операций экземпляров класса управляемых объектов, которые используют данный пакет.

Примечания

1 Операции, идентифицированные в определении класса, относятся к типам операций, определенным в ГОСТРИСО/МЭК10165-1 (Get attribute value, Replace attribute value, Replace with default value и пр.). В случае операций Actions и Notification требуются дополнительные определения для характеристики их функций, как описано в 8.10 и 8.11. Операции Create и elete специфицируются как часть шаблона связывания имен, описанного в 8.6, так как создание и удаление управляемого объекта более тесно связано соотношением вложения между старшими подчиненными объектами, чем всеми экземплярами класса управляемых объектов.

2 Отложенное связывание, т.е. присвоение дополнительных параметров действия мисообщениям класса управляемых объектов, может быть осуществлено путем включения в класс управляемых объектов пакета, который содержит (только) соответствующие действия и пакеты их новых параметров. Правила объединения, приведенные в 8.3.3 для параметров действий и сообщений, означают, что дополнительные параметры будут связаны с сообщением или действием только при реализации пакета.

8.4.2 Структура шаблона

<метка-пакета **PACKAGE**

```

[BEHAVIUR           <метка-определения-поведения
                        [, <метка-определения-поведения]*];
]
[ATTRIBUTES        <метка-атрибута список-свойств [<метка-параметра]*
                        [, <метка-атрибута список-свойств [<метка-параметра]*]*];
]
[ATTRIBUTEGRUPS   <метка-группы [<метка-атрибута]*
                        [, <метка-группы [<метка-атрибута]*]*];
]
[ACTINS           <метка-действия [<метка-параметра]*
                        [, <метка-действия [<метка-параметра]*]*];
]
[NTIFICATINS     <метка-сообщения [<метка-параметра]*
                        [, <метка-сообщения [<метка-параметра]*]*];
]

```

[REGISTERED AS идентификатор-объекта];

supporting productions

```

список-свойств- [REPLACE-WITH-DEFAULT]
                 [DEFAULTVALUE           спецификатор-значения]
                 [INITIALVALUE         спецификатор-значения]
                 [PERMITTEDVALUES     указание-типа]
                 [REQUIREDVALUES     указание-типа]
                 [получить-заменить]
                 [добавить-удалить]
                 [SET-BY-CREATE]
                 [N - M I F Y]

```

спецификатор-значения- указание-значения|

ERIVATION RULE <метка-определения-поведения

получить-заменить - **GET|REPLACE|GET-REPLACE**

добавит-удалить - **A | REMOVE | A - REMOVE**

8.4.3 Обеспечивающие определения

8.4.3.1 **BEHAVIUR** <метка-определения-поведения

[, <метка-определения-поведения]*

Конструкция **BEHAVIUR** позволяет полностью описать поведение (семантику), связанное с пакетом. Эта конструкция соотносит внешние аспекты управляемого объекта (его операции и сообщения) с его внутренними состояниями. Метка-определения-поведения идентифицирует экземпляр использования шаблона поведения. При некоторых обстоятельствах могут быть определены пакеты, в которых не требуется спецификация поведения.

8.4.3.2 **ATTRIBUTES** <метка-атрибута список-свойств
[<метка-параметра]* [, <метка-атрибута
список-свойств [<метка-параметра]*]*

Данная конструкция позволяет включать атрибуты в определение пакета. Список-свойств, который следует за каждой меткой-атрибута, определяет набор операций, которые могут осуществляться над управляемым объектом с указанием этого атрибута, и определяет значения по умолчанию, начальные, допустимые и обязательные значения, связанные с этим атрибутом.

Свойство **REPLACE-WITH-EFAULT** включается в определение, если атрибут имеет значение по умолчанию, которое может быть установлено с помощью операции *Replace with default value*.

Свойство **EFAULTVALUE** включается в определение, если атрибут имеет значение по умолчанию, которое должно использоваться для обеспечения значения атрибута в операции *Replace with default value* или должно задавать для атрибута значение по умолчанию при реализации пакета в соответствии с правилами, определенными в **ГОСТ Р ИСО/МЭК 10165-1**. Если значение по умолчанию не определено, а свойство **REPLACE-WITH-EFAULT** присутствует, то значение по умолчанию определяется способами, локальными для управляемой системы. Значение может быть задано или с помощью указания-значения, или с помощью конструкции **ERIVATIN RULE**, которая устанавливает, как может быть определено значение по умолчанию.

Свойство **INITIAL VALUE** включается в определение, если атрибут имеет обязательное начальное значение, которое должно использоваться для атрибута в момент создания. Значение может быть задано с помощью или указателя-значения, или конструкции **ERIVATIN RULE**, которая устанавливает, как может быть определено значение по умолчанию.

Если присутствует свойство **PERMITTE VALUES**, то указание-типа специфицирует ограничения на допустимые значения, которые может принимать атрибут. Указываемая спецификация должна иметь вид подтипа синтаксиса атрибута, определенного с использованием нотации АСН. I для подтипа.

Примечание 1 — Конструкция **PERMITTE VALUES** требуется только в тех определениях атрибутов, в которых необходимо задать ограничение на множество значений, допустимое синтаксисом атрибута, например при изменении существующей спецификации атрибута. Такое ограничение на множество значений атрибута должно устанавливаться только тогда, когда оно основано на ограничении наследования в семантике атрибута, а не на некоторых произвольных допущениях относительно того, что может образовывать приемлемое множество значений.

Если присутствует свойство **REQUIRE VALUES**, то указание-типа специфицирует значения, которые атрибут должен быть способен принимать. Указываемая спецификация должна иметь вид подтипа синтаксиса атрибута, определенного с использованием нотации АСН. I для подтипа.

Примечание 2 — Это свойство определяет множество значений, требуемое для соответствия. Например, управляемый объект модема может иметь атрибут скорости передачи данных с допустимыми значениями от 0 до 19,2 К; однако соответствие модема стандарту может требовать обеспечения одной конкретной скорости передачи данных из множества допустимых значений. Как в случае с конструкцией **PERMITTE VALUES**, такое ограничение на множество значений атрибута должно устанавливаться только тогда, когда оно основано на ограничении наследования в семантике атрибута, а не на некоторых произвольных допущениях относительно того, что может образовывать приемлемое множество значений.

Свойство **GET** присутствует, если значение атрибута может быть получено с помощью операции **Getattributevalue**.

Свойство **REPLACE** присутствует, если атрибут может быть установлен с помощью операций **Setattributevalue** и **Create**. Установка с помощью операции **Create** применяется только тогда, когда эта операция поддерживается связыванием имен экземпляра управляемого объекта.

Свойство **GET-REPLACE** является обозначением того, что присутствуют как свойство **GET**, так и свойство **REPLACE**.

Свойство **A** присутствует, если атрибут может быть установлен с помощью операции **Addmember**.

Свойство **RE MOVE** присутствует, если атрибут может быть установлен с помощью операции **Removemember**.

Свойство **A - R E M V E** является обозначением присутствия как свойства **A**, так и **RE MOVE**.

Свойство **SET-BY-CREATE** присутствует, если атрибут может быть установлен с помощью операции **Create**. Это свойство имеет смысл только тогда, когда операция **Create** поддерживается

связыванием имен экземпляра управляемого объекта. Так как свойство **REPLACE** присутствует, если атрибут может быть установлен с помощью операций **Set attribute value** и **Create**, то свойство **SET-BY-CREATE** не должно включаться в шаблон, если присутствует свойство **REPLACE**. Аналогично, свойство **SET-BY-CREATE** не должно включаться в шаблон, если присутствует свойство **A , R E M V E** или **A - R E M V E**. Даже когда свойство **SET-BY-CREATE** отсутствует, попытка установить значение с помощью операции **Create** может оказаться успешной.

Отсутствие свойства **REPLACE** может быть использовано для спецификации того, что атрибут не может быть изменен в экземплярах класса, но его отсутствие не исключает подклассов, в которых добавлено свойство **REPLACE**. Свойство **N - M I F Y** присутствует для того, чтобы явно установить, что атрибут не может быть изменен (доступен только для чтения) в классе, имеющем это свойство, во всех его подклассах и во всех совместимых с ним управляемых объектах (т.е. управляемых объектах, ведущих себя алломорфно этому классу). Это свойство несовместимо, следовательно, не должно присутствовать в определении класса управляемых объектов, которое имеет для того же самого атрибута любое из свойств **REPLACE**, **GET-REPLACE**, **A , R E M V E** или **A - R E M V E**.

П р и м е ч а н и я

3 Свойство **N - M I F Y** может быть совместимо со свойством **REPLACE-WITH-EFAULT**, т. к. эта операция часто используется в смысле «установить повторно», что может согласовываться с возможностями управляющего контролировать значения атрибутов.

4 До того как свойство **N - M I F Y** было добавлено в РООУ, было принято специфицировать это свойство в шаблонах **BEHAVIOUR** или в документах, на которые эти шаблоны ссылаются.

Если желательно, чтобы частью определения атрибута являлось утверждение о том, что атрибут не может быть изменен ни в каком использующем этот атрибут классе, то это ограничение должно быть установлено в шаблоне **BEHAVIOUR**, на который ссылается шаблон **ATTRIBUTE**.

Метки-параметров, если они есть, идентифицируют специфичные для класса управляемых объектов параметры ошибок, связанные с операциями управления над этим атрибутом. Сообщение о них приводятся как ошибки обработки. Синтаксис параметров ошибок определяется в указываемых шаблонах.

8.4.3.3 **ATTRIBUTEGRUPS** <метка-группы [<метка-атрибута]* [, <метка-группы [<метка-атрибута]*]*

Эта конструкция позволяет идентифицировать атрибутивные группы как часть пакета. В случае расширяемой атрибутивной группы ее исходное определение может быть расширено добавлением меток-атрибутов.

8.4.3.4 **ACTINS** <метка-действия [<метка-параметра]* [, <метка-действия [<метка-параметра]*]*

Метки-действий, если они есть, идентифицируют определения действий, которые включают в себя пакет. Определения поведения должны специфицировать результаты этих действий на управляемые объекты.

Метки-параметров, если они есть, идентифицируют специфичную для класса управляемых объектов информацию действия или параметры ответа, или специфичные для класса управляемых объектов параметры ошибок, связанные с действием. Синтаксис параметров определяется в указываемых шаблонах.

8.4.3.5 **NTIFICATIONS** <метка-сообщения [<метка-параметра]* [, <метка-сообщения [<метка-параметра]*]*

Конструкция присутствует, если в пакет включаются сообщения. Метки-сообщений идентифицируют применяемые определения сообщений. Определения поведения должны специфицировать обстоятельства, при которых эти сообщения создаются управляемыми объектами.

Метки-параметров, если они есть, идентифицируют специфичную для класса управляемых объектов информацию сообщения или параметры ответа, или специфичные для класса управляемых объектов параметры ошибок, связанные с сообщением. Здесь могут использоваться дополнительные параметры, например для заполнения поля дополнительной информации сообщения, определенное в ИСО/МЭК 10164-4. Синтаксис параметров определяется в указываемых шаблонах.

8.4.3.6 **REGISTEREAS** идентификатор-объекта

Значение идентификатора-объекта, если оно есть, обеспечивает глобально однозначный идентификатор определения пакета и регистрацию определенного пакетом группирования поведения,

атрибутов, атрибутивных групп, действий и сообщений. Значение идентификатора-объекта является тем значением, которое включается в атрибут `Packages` всех создаваемых экземпляров класса управляемых объектов в соответствии с правилами, определенными в ГОСТРИСО/МЭК10165-1. Эта конструкция обязательна, когда пакетсылается конструкция `C N I T I N A L P A C K A G E S` в шаблоне класса управляемых объектов.

8.5 Шаблон параметра

8.5.1 Обзор

Данный шаблон позволяет специфицировать зарегистрированные синтаксис параметра и соответствующее поведение, которые могут быть связаны с конкретными атрибутами, операциями и сообщениями в шаблонах пакета, атрибута, действия и сообщения, определенных в 8.4, 8.7, 8.10 и 8.11. Тип, определенный в шаблоне параметра, используется для заполнения конструкции `ANY E F I N E B U X` в ПБД административного управления, где `x`—поле ПБД, которое содержит идентификатор объекта, присвоенный параметру. Этот метод применим, например, к:

- отказам обработки;
- параметрам запросов и ответов действий;
- параметрам запросов и ответов сообщений.

Использование шаблона в каждом из этих контекстов описано в 8.5.3.

Основные элементы определения описаны ниже.

8.5.1.1 Определение контекста

Шаблон специфицирует контекст, в котором применяется параметр, а именно он устанавливает, что передает параметр в конкретном поле ПБД административного управления.

8.5.1.1.1 Информация/ответ действия, информация/ответ события, специфичная ошибка

Когда контекст недвусмысленно идентифицируется ПБД административного управления, в котором параметр передан, этот контекст может быть указан одним из пяти ключевых слов, определенных в 8.5.3.1. Контекст недвусмысленно идентифицируется ПБД административного управления только в том случае, когда конструкция `ANY E F I N E B U X` встречается в этом ПБД ровно один раз.

8.5.1.1.2 Ключевое слово контекста

Если контекст не идентифицируется недвусмысленно ПБД административного управления, в котором передается параметр, то этот контекст должен быть специфицирован ключевым словом. Ключевое слово контекста должно идентифицировать поле в ПБД административного управления, в котором может передаваться параметр.

8.5.1.1.3 Использование в других шаблонах

В таблице 16 показано, где даются ссылки на шаблон параметра.

Таблица 16 — Использование шаблона параметра

Использование	Возможные контексты
Конструкция <code>ATTRIBUTES</code> в шаблоне пакета	SPECIFIC-ERRR
Конструкция <code>ACTINS</code> в шаблоне пакета	ключевое-слово-контекста, SPECIFIC-ERRR , ACTIN-INF , ACTIN-REPLY
Конструкция <code>NTIFICATIONS</code> в шаблоне пакета	ключевое-слово-контекста, SPECIFIC-ERRR , EVENT-INF , EVENT-REPLY
Конструкция <code>CREATE</code> в шаблоне связывания имен	SPECIFIC-ERRR
Конструкция <code>ELET</code> в шаблоне связывания имен	SPECIFIC-ERRR
Шаблон атрибута	SPECIFIC-ERRR

Использование	Возможные контексты
Шаблон действия	ключевое-слово-контекста, SPECIFIC-ERRR , ACTIN-INF , ACTIN-REPLY
Шаблон сообщения	ключевое-слово-контекста, SPECIFIC-ERRR , EVENT-INF , EVENT-REPLY

При использовании в качестве квалификатора в определении пакета, параметр может быть «связан позже» с элементом, который он квалифицирует, например дополнительные параметры могут быть добавлены к ранее определенному сообщению в момент определения пакета, если синтаксис сообщения является расширяемым.

8.5.1.2 Определение синтаксиса

Шаблон позволяет связать параметр с абстрактным синтаксисом.

8.5.1.3 Указание атрибута

Вместо явного определения синтаксиса и регистрации в шаблоне параметра шаблон может специфицировать эти два элемента через ссылку на шаблон атрибута. Использование такой конструкции не влияет на смысл существующих зарегистрированных атрибутов.

8.5.1.4 Поведение

Шаблон определяет любое поведение, которое применяется к использованию параметра.

8.5.2 Структура шаблона

<метка-параметра **PARAMETER**

CNTEXT тип-контекста;

выбор-синтаксиса-или-атрибута;

[BEHAVIUR <метка-определения-поведения
[, <метка-определения-поведения]*;

]

[REGISTERED AS идентификатор-объекта];

supporting productions

тип-контекста	-	ключевое-слово-контекста ACTIN-INF ACTIN-REPLY EVENT-INF EVENT-REPLY SPECIFIC-ERRR
ключевое-слово-контекста	-	указание-типа.<идентификатор
выбор-синтаксиса-или-атрибута	-	WITHSYNTAX указание-типа ATTRIBUTE <метка-атрибута

8.5.3 Обеспечивающие определения

8.5.3.1 **CNTEXT** тип-контекста

Эта конструкция определяет контекст, в котором используется параметр, следующим образом:

- ключевое-слово-контекста: — этот выбор является ссылкой на контекст, определенный для шаблона внешним образом. Структура ссылки состоит из указания-типа последующими идентификатором, который является именем поля в ПБД административного управления, заданного указанием-типа. Следовательно, может быть использована ссылка на контекст, определенный в другом документе. Это можно использовать, например, для указания того, что параметр применяется только для конкретного поля в параметре информации события УОИУ (см. ИСО/МЭК 10164-4) или в параметре ответа действия УОИУ. Если параметр не отображается в конкретное имя поля (например, информация о событии, по определению, должна быть установлена в паре идентификатор

параметра/значение параметра), то может быть задан один из перечисленных ниже более общих контекстов:

- **ACTIN-INF**: — этот выбор определяет параметр как применяемый для представления параметров, которые могут передавать информацию действия УОИУ;
- **ACTIN-REPLY**: — этот выбор определяет параметр как применяемый для представления параметров, которые могут передавать ответ действия УОИУ;
- **EVENT-INF**: — этот выбор определяет параметр как применяемый для представления параметров, которые могут передавать информацию события УОИУ;
- **EVENT-REPLY**: — этот выбор определяет параметр как применяемый для представления параметров, которые могут передавать ответ события УОИУ;
- **SPECIFIC-ERRR**: — этот выбор определяет параметр как применяемый для представления или создания сообщений об ошибках обработки УОИУ. Когда этот выбор используется с параметрами, которые применяются для атрибутов, то определение класса управляемых объектов должно специфицировать, должны ли изменяться другие атрибуты, указанные в одном запросе замены значений, если эта ошибка происходит для одного атрибута в операции Replace attribute value или Replace with default value.

8.5.3.2 **WITHSYNTAX** Указание-типа

Данная конструкция, если она присутствует, идентифицирует тип АСН. 1 параметра при передаче в протоколе.

8.5.3.3 **ATTRIBUTE** <метка-атрибута

Данная конструкция, если она присутствует, идентифицирует шаблон атрибута, синтаксис и идентификатор объекта которого используется в качестве синтаксиса и идентификатора объекта параметра.

8.5.3.4 **BEHAVIOR** <метка-определения-поведения [, <метка-определения-поведения]*

Данная конструкция, если она присутствует, позволяет специфицировать любое поведение или семантику, связанные с данным параметром. Если используется конструкция **ATTRIBUTE**, то рассматриваемая конструкция не изменяет поведение атрибута.

8.5.3.5 **REGISTEREDAS** Идентификатор-объекта

Значение идентификатора-объекта, если оно есть, обеспечивает глобально однозначный идентификатор определения параметра. Данное значение используется в протоколе административного управления, когда необходимо идентифицировать параметр. Эта конструкция должна присутствовать только тогда, когда присутствует конструкция **WITHSYNTAX**.

8.6 Шаблон связывания имен

8.6.1 Обзор

Этот шаблон позволяет определить альтернативные структуры наименования для управляемых объектов данного класса с помощью связывания имен. Связывание имен позволяет выбрать атрибут в качестве именуемого атрибута, который будет использоваться, когда подчиненному объекту, являющемуся экземпляром заданного класса управляемых объектов, присваивается имя старшим объектом, являющимся экземпляром заданного класса управляемых объектов или класса других объектов, например класса объектов справочника.

Если используется данное связывание имен, то подчиненному объекту должен присутствовать атрибут, идентифицированный как именуемый. Именуемый атрибут используется для построения относительных отличающих имен (ОИ) подчиненных объектов этого класса. ОИ строится из идентификатора объекта, присвоенного этому типу атрибута, и значения экземпляра атрибута. Отличающее имя подчиненного объекта получается путем добавления его ОИ к отличающему имени его старшего объекта.

Связывание имен не рассматривается как часть определения классов, к которым они относятся. Данный подчиненный класс управляемых объектов может иметь несколько относящихся к нему связываний имен. Множество связываний имен определяет множество возможных именуемых взаимосвязей со старшими объектами и множество классов управляемых объектов, из которых могут реализовываться подчиненные объекты.

Связывание имен может быть определено так, что будет применяться ко всем подклассам заданного старшего класса объектов, или ко всем подклассам заданного подчиненного класса объектов, или к тем и другим.

Примечание — Связывание имен для класса управляемых объектов может быть установлено после спецификации самого класса.

8.6.2 Структура шаблона

<метка-связывание-имен**NAME BINING**
SUB RINATE BJECT CLASS <метка-класса [**AN SUBCLASSES**];
NAME BY SUPERIR BJECT CLASS <метка-класса [**AN SUBCLASSES**];
WITHATTRIBUTE <метка-атрибута;
BEHAVIUR <метка-определения-поведения
 [, <метка-определения-поведения]*;
]
CREATE [модификатор-создания[, модификатор-создания]]
 [<метка-параметра]*;
]
DELETE [модификатор-удаления]
 [<метка-параметра]*;
]

REGISTERE ASидентификатор-объекта;

supporting productions

модификатор-создания - **WITH-REFERENCE-BJECT |**
WITH-AUTMATIC-INSTANCE-NAMING
 модификатор-удаления - **NLY-IF-N - CNTAINE - BJECTS |**
ELETES - CNTAINE - BJECTS

8.6.3 Обеспечивающие определения

8.6.3.1 **SUB RINATE BJECT CLASS**<метка-класса
 [**AN SUBCLASSES**]

Конструкция определяет класс управляемых объектов, экземплярам которого могут присваиваться имена экземплярам класса объектов, определенного конструкцией **NAME BY SUPERIR BJECT CLASS**. Имя экземпляра этого подчиненного класса объектов образуется сцеплением отличающего имени гостаршего объекта с относительным отличающим именем подчиненного объекта. Если задано **AN SUBCLASSES**, то это связывание имен применяется для всех подклассов заданного класса управляемых объектов.

8.6.3.2 **NAME BY SUPERIR BJECT CLASS**<метка-класса
 [**AN SUBCLASSES**]

Конструкция определяет класс управляемых объектов или класс других объектов, например класс объектов справочника, экземпляры которого могут присваивать имена экземплярам класса управляемых объектов, определенного конструкцией **SUB RINATE BJECT CLASS**. Если задано **AN SUBCLASSES**, то это связывание имен применяется для всех подклассов заданного класса управляемых объектов.

8.6.3.3 **WITHATTRIBUTE**<метка-атрибута

Эта конструкция определяет атрибут, который должен использоваться в контексте рассматриваемого связывания имен для построения относительного отличающего имени экземпляра класса управляемых объектов, определенного конструкцией **SUB RINATE BJECT CLASS**. Значения этого атрибута должны представляться типом данных с единственным значением, удовлетворяющим ограничениям ГОСТРИСО/МЭК10165-1. Если нет подходящего атрибута для использования в качестве именуемого, то проектировщикам управляемых объектов рекомендуется обеспечивать управляющий атрибут типа **GraphicString**.

8.6.3.4 **BEHAVIUR**<метка-определения-поведения
 [, <метка-определения-поведения]*

Если она присутствует, эта конструкция позволяет определить любые конфликты поведения, возникающие вследствие связывания имен. Метка-определения-поведения идентифицирует рассматриваемое определение поведения.

Примечание — Эта конструкция предназначена для использования в качестве средства описания поведения, специфичного для связывания имен. Любое поведение, которое применимо ко всем возможным экземплярам класса управляемых объектов, должно быть определено как часть поведения, указанного в шаблоне пакета, определяющего класс управляемых объектов.

8.6.3.5 **CREATE**[модификатор-создания
 [, модификатор-создания]][<метка-параметра]*

Конструкция присутствует, если допускается создание новых экземпляров класса управляемых объектов, указанного конструкцией **SUB R I N A T E B J E C T C L A S S**, в контексте данного связывания имен с помощью операции административного управления системой. Значения модификаторов-создания специфицируют опции, доступные при создании. Допустимыми значениями являются следующие:

- **WITH-REFERENCE-BJECT**: при создании может быть задан ссылка на управляемый объект как источник значений по умолчанию для спецификации выбора условных пакетов;

- **WITH-AUTOMATIC-INSTANCE-NAMING**: можно опустить в запросе создания спецификацию имени экземпляра нового управляемого объекта.

Определения поведения должны специфицировать, как должны выбираться действия, когда выбраны связывания имен, которые могут применяться к новому управляемому объекту.

Источники начальных значений атрибутов, используемых в момент создания управляемого объекта, и соответствующие правила предпочтения определены в ГОСТРИСО/МЭК 10165-1.

Метки-параметров, если они есть, идентифицируют параметры специфичных для связывания имен ошибок, связанных с операцией **Create**. Они сообщаются как ошибка во времени обработки. Синтаксис параметров ошибок определяется указываемыми шаблонами.

8.6.3.6 **ELETE**[модификатор-удаления][<метка-параметра]*

Конструкция присутствует, если допускается удаление экземпляров класса управляемых объектов, указанного конструкцией **SUB R I N A T E B J E C T C L A S S**, в контексте данного связывания имен. Модификатор-удаления, если он есть, указывает поведение при удалении управляемого объекта этого класса. Допустимыми значениями являются следующие:

- **NLY-IF-N - C N T A I N E - B J E C T S**: все вмещаемые управляемые объекты должны быть явно удалены с помощью операций административного управления до удаления вмещающего управляемого объекта, т. е. запрос операции **delete** приведет к ошибке, если имеются вмещаемые управляемые объекты;

- **ELETES - C N T A I N E - B J E C T S**: если операция **delete** применяется к управляемому объекту, для которого задан этот модификатор, то запрос операции **delete** приведет к отказу, если какой-либо прямо или косвенно вмещаемый управляемый объект имеет модификатор **NLY-IF-N - C N T A I N E - B J E C T S** и содержит управляемые объекты; в противном случае успешный запрос операции **delete** приведет к удалению всех вмещаемых управляемых объектов.

Другие правила, описывающие поведение относительно удаления вмещаемых управляемых объектов, могут быть специфицированы в конструкции **B E H A V I U R**.

Примечание — Учитывая, что модификатор **ELETES - C N T A I N E - B J E C T S** допускает удаление управляемого объекта независимо от того, содержит ли он другие управляемые объекты, предпочтительнее использовать модификатор **NLY-IF-N - C N T A I N E - B J E C T S**, если есть какие-либо сомнения относительно того, какой модификатор лучше подходит.

Если существуют ограничения на удаление относительно других взаимосвязей или условий, которые являются родовыми для класса управляемых объектов, то они должны быть специфицированы как часть поведения класса управляемых объектов.

Метки-параметров, если они есть, идентифицируют параметры специфичных для связывания имен ошибок, связанных с операцией **delete**. Они сообщаются как ошибка во времени обработки. Синтаксис параметров ошибок определяется указываемыми шаблонами.

8.6.3.7 **REGISTERED AS** идентификатор-объекта;

Значение идентификатора-объекта, если оно есть, обеспечивает глобально однозначный идентификатор определения связывания имен. Данное значение используется для идентификации связывания имен в целях административного управления.

8.7 Шаблон атрибута

8.7.1 Обзор

Данный шаблон используется для определения отдельных типов атрибутов. Эти определения могут быть в дальнейшем объединены в шаблон атрибутивной группы. Основные элементы определения описаны ниже.

8.7.1.1 Получение

Определение типа атрибута может изменять или ограничивать определение другого типа атрибута.

8.7.1.2 Синтаксис атрибута

Определение типа атрибута должно включать в себя определение синтаксиса, который должен использоваться для выражения значений атрибута в протоколе административного управления. Это достигается с помощью ссылки на определение типа АСН.1. Определение синтаксиса атрибута указывает, является ли значение атрибута одно- или многозначным. Если базовым типом является SET OF, то тип атрибута — многозначный, в противном случае — однозначный.

8.7.1.3 Согласование значений

Определение типа атрибута может включать в себя допустимые способы, которыми может быть проверено значение экземпляра типа, т.е. атрибут может быть проверен на равенство, размеры и т.п. Для некоторых типов атрибутов согласование значения может потребовать, как часть определения поведения атрибута, спецификацию того, как определены используемые правила согласования. Отсутствие правил согласования в определении атрибута подразумевает, что согласование значений не определено.

8.7.1.4 Поведение

Определение атрибута может включать в себя определение специфичного для атрибута поведения, т.е. поведения, которое применяется для типа атрибута, независимо от того, какой класс управляемых объектов содержит экземпляр типа атрибута.

8.7.1.5 Идентификатор атрибута

Значение идентификатора объекта должно быть выделено для каждого атрибута, который должен включаться в определение класса управляемых объектов. Это значение используется в протоколе административного управления для идентификации атрибута.

8.7.1.6 Параметры

Определение атрибута может идентифицировать параметры специфичные для атрибута оболочки, связанные с операциями управления над атрибутом этого типа.

8.7.2 Структура шаблона

<метка-атрибута

ATTRIBUTE

получен-из-или-синтаксис;

[**MATCHES FR** квалификатор

[, квалификатор]*;

]

[**BEHAVIUR** <метка-определения-поведения

[, <метка-определения-поведения]*;

]

[**PARAMETERS** <метка-параметра

[, <метка-параметра]*;

]

[**REGISTERED AS** идентификатор-объекта];

supporting productions

квалификатор

- **EQUALITY | RERING | SUBSTRINGS | SET-CMPARISN | SET-INTERSECTIN**

получен-из-или-синтаксис

- **ERIVE FR M** <метка-атрибута |

WITH ATTRIBUTESYNTAX указание-типа

8.7.3 Обеспечивающие определения

8.7.3.1 **ERIVE FR M** <метка-атрибута

Если эта конструкция присутствует, то определение атрибута в качестве исходной точки принимает все элементы определения, указываемые меткой-атрибута, включая те, которые, в свою очередь, могут быть получены из других определений атрибутов. В этом случае правила интерпретации результата присутствия любого другого элемента шаблона атрибута следующие:

- **MATCHES FR**: результирующий набор правил согласования является логическим ИЛИ правил согласования, задаваемых этой конструкцией, совместно использованными правилами согласования;

- **BEHAVIUR**: — расширяет любые заимствованные правила согласования;

- **REGISTERED AS**: — заменяет любую заимствованную регистрацию.

Этот метод вывода одного атрибута из другого позволяет:

- определять атрибут на основе другого существующего определения атрибута;

- добавлять дополнительные ограничения к существующему определению атрибута.

8.7.3.2 **WITH ATTRIBUTE SYNTAX** Указание-типа

Эта конструкция, присутствующая только в том случае, когда отсутствует конструкция **ERIVE FR M**, идентифицирует тип данных АСН.1, который описывает, как экземпляры значений атрибута передаются в протоколе.

Тип данных АСН.1 также определяет тип данных самого атрибута. Если базовым типом синтаксиса является «множество-из», то атрибут может иметь несколько значений. Все остальные типы данных АСН.1, включая типы «множество», «последовательность» и «последовательность-из», определяют типы атрибутов единственным значением.

8.7.3.3 **MATCHES F R** квалификатор[, квалификатор]*

Эта конструкция определяет типы проверок, которые могут применяться к значению атрибута как часть фильтра операции. Согласование на наличие атрибута неявно подразумевается для всех атрибутов. Все согласования других типов, если эта конструкция отсутствует, неопределены, следовательно, недопустимы для атрибута. Варианты квалификаторов следующие:

- **EQUALITY**: — значение атрибута, если оно есть, может быть проверено на равенство заданному значению;
- **R ER I N G**: — значение атрибута, если оно есть, можно сравнить с заданным значением для определения большего из них;
- **SUBSTRINGS**: — значение атрибута, если оно есть, можно сравнить с заданным значением для определения, входит оно или нет в значение атрибута;
- **SET-C M P A R I S N**: — значение атрибута, если оно есть, можно сравнить с заданным значением для определения соотношения супермножество/подмножество между этими значениями;
- **SET-INTERSECT I N**: — значение атрибута, если оно есть, можно сравнить с заданным значением для нахождения непустого пересечения этих двух значений.

8.7.3.4 **BEHAVIUR** <метка-определения-поведения

[, <метка-определения-поведения]*

Любое поведение, которое является родовым для данного типа атрибута, может быть определено с помощью этой конструкции. Определение поведения должно включать в себя любые дополнительные спецификации, которые требуются для определения того, как выбранное множество правил согласования применяется к определению атрибута. Поведение, которое является специфичным для класса управляемых объектов, определяется в конструкции **BEHAVIOUR** шаблона пакета.

8.7.3.5 **PARAMETERS** <метка-параметра[, <метка-параметра]*

Метки-параметров позволяют связать параметры поведения типа атрибута для определения отказов обработки. Например, при некоторых обстоятельствах тип атрибута может продемонстрировать ошибку «нарушение ограничения». Параметр, дающий информацию о такой ошибке, может быть определен, используя **C N T E X T S P E C I F I C - E R R R** в шаблоне параметра, и указани из шаблона атрибута.

8.7.3.6 **REGISTERED AS** идентификатор-объекта

Значение идентификатора-объекта, если оно есть, обеспечивает глобально однозначный идентификатор определения атрибута, которое включает в себя все элементы, прямо или косвенно указанные в конструкции **ERIVE FR M**, **WITH ATTRIBUTE SYNTAX**, **MATCHES F R** и **BEHAVIUR**. Это значение используется в протоколе административного управления, когда необходимо идентифицировать тип атрибута. Если эта конструкция опущена, то на определение атрибута нельзя сослаться во определении класса управляемых объектов. Когда определение атрибута выводится из существующего определения атрибута, которое содержит конструкцию **REGISTERED AS**, значение идентификатора-объекта, присвоенное существующему определению, не является допустимым идентификатором для выводимого определения. Следовательно, конструкция **REGISTERED AS** должна быть включена в выводимое определение, если на него нужно сослаться из определения класса управляемых объектов.

8.8 Шаблон атрибутивной группы

8.8.1 Обзор

Данный шаблон позволяет определять атрибутивные группы; такое группирование применяется в ситуациях, когда желательно работать совокупностью атрибутов, которые являются членами группы. Определения поведения для конкретного класса управляемых объектов устанавливаются с помощью операций получения значения атрибута и замены значением по умолчанию в тех случаях, когда операции применяются к атрибутивным группам. Каждый член группы сам должен быть определен как одно- или многозначный тип атрибута.

Шаблон атрибутивной группы определяет минимальный набор атрибутов, которые образуют группу, изначение идентификатора объекта, которое используется для наименования группы. Каждое определение класса управляемых объектов, которое ссылается на атрибутивную группу, может расширить группу, добавив новых членов, если только группа не была определена как фиксированная. Такие расширения применяются только для экземпляров того класса управляемых объектов, в котором расширение определено. Атрибуты, идентифицированные в шаблоне атрибутивной группы, определяют минимальный состав группы во всех определениях классов управляемых объектов, ссылающихся на эту группу.

Если в определении класса управляемых объектов присутствует расширяемая атрибутивная группа, то все атрибуты, включенные в группу либо в тело шаблона атрибутивной группы, либо добавленные в определение класса управляемых объектов, должны присутствовать в пакете, который ссылается на группу, или в одном из обязательных пакетов этого класса.

Если в определении класса управляемых объектов присутствует фиксированная атрибутивная группа, то все атрибуты, включенные в группу, должны присутствовать в пакете, который ссылается на группу.

8.8.2 Структура шаблона

```
<метка-группы      ATTRIBUTEGRUPS
  [GROUP ELEMENTS <метка-атрибута
                    [, <метка-атрибута]*;
  ]
  [FIXE;
  ]
  [ESCRIP TI N      выделенная-строка;
  ]
```

REGISTEREAS идентификатор-объекта;

8.8.3 Обеспечивающие определения

8.8.3.1 **GRUPELEMENTS** <метка-атрибута [, <метка-атрибута]*

Эта конструкция, если она есть, определяет набор меток-атрибутов, которые идентифицируют отдельные атрибуты, образующие те элементы атрибутивной группы, которые должны присутствовать во всех экземплярах этой группы; каждый из этих элементов должен быть определен с помощью шаблона атрибута. Определения поведения для конкретного класса управляемых объектов устанавливаются с помощью операций получения значения атрибута и замены значения по умолчанию в тех случаях, когда операции применяются к атрибутивным группам.

Примечание — Это не подразумевает, что существует спецификация поведения самой атрибутивной группы, которая не применяется к отдельным атрибутам.

Все атрибуты в группе должны быть членами определения класса управляемых объектов, ссылающегося на группу, т. е. каждый атрибут, который является членом группы для данного класса управляемых объектов, должен быть указан в конструкции **ATTRIBUTE**S одного или нескольких пакетов, на которые ссылается определение класса управляемых объектов.

8.8.3.2 **FIXE**

Эта конструкция, если она есть, указывает, что атрибутивная группа определяется как фиксированная.

8.8.3.3 **ESCRIP TI N** выделенная-строка

Эта конструкция позволяет описать семантику группы, например: «Группа всех атрибутов состояний управляемого объекта». Не устанавливается никаких ограничений на наборы символов, используемые в данной конструкции, и не определяется какая-либо структура внутри нее.

Данная конструкция не должна использоваться как способ определения поведения группы или ее членов.

8.8.3.4 **REGISTEREAS** идентификатор-объекта

Значение идентификатора-объекта обеспечивает глобально однозначный идентификатор определения атрибутивной группы. Это значение используется в протоколе административного управления, когда необходимо идентифицировать атрибутивную группу. Аtribuтивная группа, идентифицированная этим значением идентификатора-объекта в контексте управляемого объекта, включает в себя все атрибуты, определенные в теле шаблона атрибутивной группы, с учетом для рас-

ширяемых групп всех атрибутов, добавленных к группам вследствие определения элементов шаблона класса управляемых объектов, который применяется при реализации управляемого объекта.

Примечание — Шаблон атрибутивной группы определяет набор атрибутов (он может быть пустым), которые всегда являются членами группы. В случае расширяемой атрибутивной группы этот набор может быть расширен конструкцией `ATTRIBUTE GROUPS` в шаблоне пакета в интересах конкретных определений классов управляемых объектов. Этот метод подходит тогда, когда желательно определить атрибутивную группу, члены которой имеют некоторую общую семантику (например, «атрибуты состояний»), но число атрибутов в момент реализации; или когда в нескольких классах управляемых объектов требуются различные группировки атрибутов с одной и той же семантикой. В общем случае можно определить состав расширяемой атрибутивной группы в управляемом объекте только в момент реализации, когда известно, какие пакеты и, следовательно, какие атрибуты должны реализовываться.

8.9 Шаблон поведения

8.9.1 Обзор

Данный шаблон используется для определения поведения классов управляемых объектов, связанных имен, параметров, атрибутов, действий и сообщений. Шаблон поведения предназначен для того, чтобы обеспечивать расширения, но спецификации поведения не должны изменять ранее определенную информацию. Если информация оставлена неопределенной, то в определении поведения должно быть явно указано, что именно неопределено.

Примечания

1 Шаблоны поведения должны использоваться для выражения семантики, которая не полностью описана в других шаблонах. В частности, авторы определений не должны полагаться на метки для выражения семантики.

2 Утверждения о поведении должны быть выражены в терминах управляемых объектов того класса, определение которого содержит эти утверждения.

8.9.2 Структура шаблона

<метка-определения-поведения **BEHAVIOR**

EFINE AS

выделенная-строка;

8.9.3 Обеспечивающие определения

8.9.3.1 **EFINE AS** выделенная-строка

Текст, содержащийся в выделенной-строке, дает определение поведения класса управляемых объектов или соответствующему связыванию имен, параметров, атрибутов, действий или сообщений. Это определение может быть задокументировано на естественном языке или с использованием формальных методов описания. Текст может быть (текстовой) ссылкой на разделы или подразделы некоторого документа или стандарта. Не устанавливаются никаких ограничений на наборы символов, используемые для представления выделенной-строки, и не определяется какая-либо структура в этом тексте.

8.10 Шаблон действия

8.10.1 Обзор

Этот шаблон используется для определения поведения и синтаксисов, связанных с конкретным типом действия. Типы действий, определенные с помощью этого шаблона, могут быть переданы услугой **M-ACTIN**, определенной в ГОСТРИСО/МЭК 9595. Ниже описаны основные элементы определения.

8.10.1.1 Поведение

Определение типа действия должно специфицировать функции действия в терминах воздействий, которые оно оказывает на классы управляемых объектов. Когда действие может применяться к нескольким классам управляемых объектов, описание поведения должно ограничиваться теми характеристиками, которые являются общими для управляемых объектов всех классов; относящееся к этому действию поведение, специфичное для класса управляемых объектов, описывается как часть определения самого класса.

8.10.1.2 Режим работы

Определение типа действия должно указывать, является ли действие всегда подтверждаемым или оно может быть подтверждаемым или неподтверждаемым по усмотрению управляющего.

8.10.1.3 Абстрактный синтаксис

Определение типа действия должно специфицировать все синтаксисы, которые могут использоваться для передачи информации действия и параметров ответа действия услуге

М-АСТIN, определенной ГОСТРИСО/МЭК9595. Синтаксисы определяются помощью типов данных АСН.1.

Примечания

1 Если только нет намерения специально предотвратить последующие расширения аргументов действий, то рекомендуется, чтобы синтаксисы информации и ответа действия определялись расширяемым образом путем включения в качестве факультативного поля типа АСН.1 **SET F Management Extension** (определенного в ГОСТРИСО/МЭК10165-2).

2 Рекомендуется, чтобы базовым типом данных, выбранным для синтаксисов информации и ответа, был тип **SEQUENCE**.

8.10.1.4 Идентификаторы действий

Значение идентификатора объекта, связанного с определением типа действия, используется для идентификации этого типа в протоколе административного управления.

8.10.1.5 Параметры

Определение типа действия может идентифицировать параметры информации действия, ответа действия или специфичных ошибок, связанных с типом действия.

8.10.2 Структура шаблона

<метка-действия **АСТIN**

```
[ BEHAVIUR <метка-определения-поведения
    [, <метка-определения-поведения]*;
]
[ M E C N F I R M E ;
]
[ PARAMETERS <метка-параметра
    [, <метка-параметра]*;
]
[ WITH INFORMATION SYNTAX указание-типа;
]
[ WITH REPLY SYNTAX указание-типа;
]
```

REGISTERED AS идентификатор-объекта;

8.10.3 Обеспечивающие определения

8.10.3.1 **BEHAVIUR** <метка-определения-поведения
[, <метка-определения-поведения]*

Эта конструкция, когда присутствует, определяет поведение действия, параметры, которые должны быть определены вместе с действием, результаты, к которым может привести действие, и их смысл. Метки-определений-поведения указывают на описания поведения, определенные с помощью шаблона поведения.

8.10.3.2 **M E C N F I R M E**

Эта конструкция, если присутствует, указывает, что действие должно осуществляться в подтверждаемом режиме. Если конструкция отсутствует, то действие может осуществляться в подтверждаемом или неподтверждаемом режиме по усмотрению управляющего.

8.10.3.3 **PARAMETERS** <метка-параметра [, <метка-параметра]*

Метки-параметров идентифицируют параметры информации или ответа действия, а также отказы обработки, связанные с типом действия. Пример см. в А.7.

8.10.3.4 **WITH INFORMATION SYNTAX** указание-типа

Если эта конструкция присутствует, то указание-типа идентифицирует тип данных АСН.1, описывающий структуру параметра информации действия, которая передается в протоколе административного управления. Если эта конструкция отсутствует, то нет никакой специфичной информации, связанной с вызовом действия.

8.10.3.5 **WITH REPLY SYNTAX** указание-типа

Если эта конструкция присутствует, то указание-типа идентифицирует тип данных АСН.1, описывающий структуру параметра ответа действия, которая передается в протоколе административного управления. Если эта конструкция отсутствует, то нет никакой специфичной информации, связанной с ответом действия.

8.10.3.6 **REGISTERE AS**Идентификатор-объекта

Значение идентификатора-объекта обеспечивает глобально однозначный идентификатор определения типа действия. Это значение используется в протоколе административного управления, когда необходимо идентифицировать тип действия.

8.11 Шаблон сообщения

8.11.1 Обзор

Данный шаблон используется для определения поведения и синтаксисов, связанных с конкретным типом сообщения. Типы сообщений, определенных с помощью этого шаблона, могут передаваться в отчетах о событиях с помощью услуги **M-EVENT-REPORT**, определенной в ГОСТРИСО/МЭК9595. Ниже описаны основные элементы определения.

8.11.1.1 Поведение

Определение типа сообщения должно специфицировать обстоятельства, при которых создается сообщение данного типа.

8.11.1.2 Абстрактный синтаксис

Определение типа сообщения должно специфицировать все абстрактные синтаксисы, которые могут использоваться для выражения параметров информации и ответа события в услуге **M-EVENT-REPORT**, определенной в ГОСТРИСО/МЭК9595. Шаблон также допускает распределение значений атрибутов по полям синтаксиса.

Примечания

1 Если только нет намерения специально предотвратить последующие расширения аргументов сообщений, то рекомендуется, чтобы синтаксис информации и ответа сообщения определялись с расширяемым образом путем включения в качестве факультативного поля типа АСН.1 **SETFManagementExtension** (определено в ГОСТРИСО/МЭК10165-2).

2 Рекомендуется, чтобы базовым типом данных, выбранным для синтаксисов информации и ответа, был тип **SEQUENCE**.

8.11.1.3 Имя сообщения

Значение идентификатора объекта, связанного с определением сообщения, используется для идентификации типа события в протоколе административного управления.

8.11.1.4 Параметры

Определение типа сообщения может идентифицировать параметры информации и события, ответа события или специфичных ошибок, связанных с типом сообщения.

8.11.2 Структура шаблона

```

<метка-сообщения      NTIFICATION
  [BEHAVIUR <метка-определения-поведения
    [, <метка-определения-поведения]*;
  ]
  [PARAMETERS <метка-параметра
    [, <метка-параметра]*;
  ]
  [WITHINFRMATIN SYNTAX указание-типа
    [AN ATTRIBUTE IS <имя-поля <метка-атрибута
      [, <имя-поля <метка-атрибута]*
    ];
  ]
  [WITHREPLY SYNTAX указание-типа;
  ]

```

REGISTERE ASИдентификатор-объекта;

8.11.3 Обеспечивающие определения

8.11.3.1 **BEHAVIUR** <метка-определения-поведения
[, <метка-определения-поведения]*

Эта конструкция, когда присутствует, определяет поведение сообщения, данные, которые должны быть определены вместе с сообщением, результаты, к которым может привести сообщение, и их смысл. Метки-определений-поведения указывают на описания поведения, определенные с помощью шаблона поведения.

8.11.3.2 **PARAMETERS** <метка-параметра[, <метка-параметра]*

Метки-параметров идентифицируют параметры информации или от ответа события, а также от казы обработки, связанные с типом сообщения. Пример см. в А.8.

8.11.3.3 **WITH INFRMATION SYNTAX** Указание-типа

[**AN ATTRIBUTE IS** <имя-поля <метка-атрибута
[, <имя-поля <метка-атрибута]*]

Если эта конструкция присутствует, то указание-типа идентифицирует тип данных АСН.1, описывающий структуру параметра информации сообщения, которая передается в протоколе административного управления, и позволяет связать идентификаторы атрибутов с именами полей в абстрактном синтаксисе. Если эта конструкция отсутствует, то нет никакой специфичной информации, связанной с вызовом сообщения. Если присутствует конструкция **AN ATTRIBUTE IS**, то имя-поля должно быть меткой, определенной в абстрактном синтаксисе, на который ссылается указание-типа. Тип данных, помеченный именем-поля, используется для передачи значений атрибута, указанного меткой-атрибута. Тип данных АСН.1 атрибута должен быть тем же, что и указанный именем-поля.

Никакие метки внутри типов **SET** Фил и **SEQUENCE** F не могут использоваться в качестве имени-поля, так как метки внутри подобных повторяющихся конструкций не позволяют недвусмысленно ссылаться на единичные экземпляры типа данных. Аналогично никакие метки компонентов типов **CHICE**, **SET** или **SEQUENCE** не могут использоваться в качестве имени-поля, если помеченные компоненты неоднократно встречаются в определении типа.

8.11.3.4 **WITH REPLY SYNTAX** Указание-типа

Если эта конструкция присутствует, то указание-типа идентифицирует тип данных АСН.1, который описывает структуру параметра ответа сообщения, которая передается в протоколе административного управления. Если эта конструкция отсутствует, то нет никакой специфичной информации, связанной с ответом сообщения.

Синтаксис ответа используется, когда сообщение отправляется в подтверждаемом режиме услуги УОИУМ-**EVENT-REPORT**. Подтверждение события не возвращается управляемому объекту. Решение об отправке сообщения в подтверждаемом или не подтверждаемом режиме является вопросом соответствующего агента, который принимает решение на основе политики, связанной с управляющим. Когда конструкция **WITH REPLY SYNTAX** опущена в определении сообщения, но сообщение отправлено в подтверждаемом режиме, то подтверждение не будет содержать информации ответа.

8.11.3.5 **REGISTERED AS** идентификатор-объекта

Значение идентификатора-объекта обеспечивает глобально однозначный идентификатор определения типа сообщения. Это значение используется в протоколе административного управления, когда необходимо идентифицировать тип сообщения.

9 Руководство по разработке эквивалентных модулей АСН.1:1994 и АСН.1:1990

Возможна разработка стандартов на основе АСН.1:1994 (ИСО/МЭК 8824-1). Для того чтобы можно было использовать АСН.1:1994, рекомендуется разрабатывать эквивалентный нормативный модуль АСН.1:1990 (ГОСТРИСО/МЭК 8824) со следующими свойствами:

- 1) он должен иметь тот же самый идентификатор объекта, что и модуль АСН.1:1994;
- 2) он является нормативным, но стандарт устанавливает, что в случае несогласованности между модулями АСН.1:1990 и АСН.1:1994, предпочтение имеет последний из них;
- 3) стандарт устанавливает, что использование АСН.1:1990 сохраняется так долго, как это будет необходимо.

Примечание 1 — Правилами ИСО/МЭК СТК1/ПК21 установлен периодический (один раз в год) обзор целью обновления международных стандартов АСН.1:1990*. Национальным органам по стандартизации рекомендовано учитывать это при пересмотре стандартов АСН.1:1990. Тем самым обеспечивается, что стандарты АСН.1:1990 будут сохраняться так долго, как это необходимо.

* ИСО/МЭК СТК1/ПК21 подтвердил продолжение действия стандартов АСН.1:1990 по соображениям соответствия и переносимости. ПК21 потребовал от своих рабочих групп продолжать поддерживать эти стандарты. Соответствующая резолюция ПК21 будет приниматься на каждом заседании ПК21 (в настоящее время — ежегодно).

Для уменьшения количества ошибок рекомендуется, чтобы модуль АСН.1:1990 генерировался в результате машинного преобразования АСН.1:1994, так как это преобразование легко осуществить автоматически.

Примечание 2 — Если для преобразования АСН.1:1994 в АСН.1:1990 желательно использовать коммерческое средство (например средство АСН.1 поставщика ХХХ), то рекомендуется в начале сгенерированного кода добавить комментарий, который гласит приблизительно следующее:

-- Использовано средство ХХХ АСН.1 --
 -- для преобразования АСН.1:1994 в АСН.1:1990 --
 и примечание:

«Примечание — Хотя ИСО не отдает предпочтения ни одному программному средству, ХХХ АСН.1 позволяет преобразовать АСН.1:1994 в АСН.1:1990.»

Следует учитывать, что проблем можно избежать только в том случае, когда используется общее подмножество АСН.1:1990 и АСН.1:1994. В таком случае стандарт следует включать только модуль АСН.1:1994.

9.1 Руководство

Рекомендуется придерживаться следующих правил.

1) Модули АСН.1:1990 и АСН.1:1994 должны ссылаться на одни и те же документы административного управления системы. Требуется, чтобы данный модуль полностью соответствовал либо АСН.1:1990, либо АСН.1:1994, а директивы, определенные в разделе 10, используются для идентификации того, какая версия нотации используется в конкретном модуле.

2) Указания типов и значений могут быть импортированы в модуль АСН.1:1994 из модуля АСН.1:1990 за следующими исключениями:

а) АСН.1:1990 MACRO не может быть импортирован в модуль АСН.1:1994; следовательно, невозможно создать экземпляр MACR в модуле АСН.1:1994.

б) Идентификаторы значений SET, SEQUENCE и CHICE.

3) Указания типов и значений могут быть импортированы в модуль АСН.1:1990 из модуля АСН.1:1994 за следующим исключением:

типы АСН.1:1994 CHARACTERSTRING, BMPString, UniversalString, EMBE E PV не могут быть импортированы. Так как в АСН.1:1990 нет эквивалентов для этих типов АСН.1:1994, то их использование не рекомендуется в тех модулях АСН.1:1994, для которых требуются эквивалентные модули АСН.1:1990. По тем же причинам запрещается использование типа АСН.1:1994 Tuple в тех модулях АСН.1:1994, для которых требуются эквивалентные модули АСН.1:1990*.

Примечание 1 — Если следовать предлагаемому руководству, то противоречия при импорте указаний типов и значений из одной версии АСН.1 в другую не возникает, т.к. эквивалентные конструкции существуют в любом случае.

4) Для определений АСН.1 классов информационных объектов, которые импортируются в модуль АСН.1:1994, следует использовать следующий модуль АСН.1:1994:

```
-- < Модуль АСН.1 версии 1994 года SMModule
-- < {joint-iso-itu-tms(9)smi(1)part4(4)asn1Module(2)2} - -
SMModule{joint-iso-itu-tms(9)smi(1)part4(4)asn1Module(2)2}
EFINITINS: : = BEGIN
REGISTERE-AS: : = TYPE-IDENTIFIER
-- TYPE-IDENTIFIER определен в ГОСТРИСО/МЭК8824—1, доступен в любом модуле без
-- его импорта и определен как:
-- TYPE-IDENTIFIER: : = CLASS
-- {
-- &id BJECTIDENTIFIERUNIQUE,
-- &Type
-- }
```

* Tuple является именем для продукции АСН.1:1994, которая позволяет вставлять управляющие символы в нотацию значения IA5String, чего нельзя сделать в АСН.1:1990. Например, ... **greetingsIA5String**: : = {«hello», сг, «there»} вставляет возврат каретки между «hello» и «there» (сг импортировано из модуля, определенного в ИСО/МЭК8824-1, и эквивалентно литералу «возврат каретки»).

```

- - WITH SYNTAX { &Type IDENTIFIER BY &id }
  INF-REPLY-IDENTIFIER ::= CLASS
  {
    &Info PTINAL,
    &Reply PTINAL,
    &registeredAs BJECTIDENTIFIERUNIQUE
  }
  WITH SYNTAX { INF &Info REPLY &Reply IDENTIFIER BY &registeredAs }
  RegisteredAsTable REGISTRE-AS ::= { ... }
  InfoReplyTable INF-REPLY-IDENTIFIER ::= { ... }
- - RegisteredAsTable должна быть заполнена шаблонами POYO
- - ATTRIBUTE и PARAMETER.
-- InfoReplyTable должна быть заполнена шаблонами POYO
-- ACTION и NOTIFICATION.
  EN
  5) Модули АСН.1:1994 определяются как в следующем примере:
-- < Модуль АСН.1 версии 1994 года ExampleModule - -
  ExampleModule { -- здесь должен быть допустимый идентификатор объекта --
    EFINITINS ::= BEGIN
    IMP RTS
    REGISTRE-AS,
    INF-REPLY-IDENTIFIER,
    RegisteredAsTable,
    InfoReplyTable
    FRMSMModule { joint-iso-itu-tms (9) smi (1) part4 (4) asn1 Module (2) 2 };
    Foo ::= SEQUENCE {
      id1 REGISTRE-AS.&id({RegisteredAsTable}),
      syntax1 REGISTRE-AS.&Type({RegisteredAsTable}{@id1})
    }
    Bar ::= SEQUENCE {
      id2 REGISTRE-AS.&id({RegisteredAsTable}),
      syntax2 SEQUENCE FREGISTRE-AS.&Type({RegisteredAsTable}{@id2})
    }
    firstExtensionId BJECTIDENTIFIER ::= { 1 3 17 103 10 1 }
    firstExtensionInfo ::= PrintableString
  }
-- Иллюстрирует использование содержащегося подтипа, ограничивающего открытый тип. --
  FooBar ::= Foo (WITH COMPONENTS {
    id1 (firstExtensionId),
    syntax1 (FirstExtensionInfo)
  })
  EN

```

Так как POYO используется вместе с классом информационных объектов АСН.1 REGISTRE-AS, то FooBar является дублированием информации. А именно, спецификация POYO плюс REGISTRE-AS АСН.1 эквивалентно ограничению внутреннего типа Foo. FooBar просто иллюстрирует, что открытый тип может быть ограничен до любого типа, а ANY/ANY EFINE BY не может быть ограничен в АСН.1:1990 до типа, отличного от ANY/ANY EFINE BY. Эта конструкция показывает, как отображать ограниченный таким образом тип АСН.1:1994 в комментарии в АСН.1:1990.

6) Модули АСН.1:1994 преобразуются в модули АСН.1:1990 последующими инструкциями.

а) Удалить часть утверждения IMP RTS, ссылающуюся на модуль SMModule. Это позволит избавиться от импортированных определений классов информационных объектов REGISTRE-AS и INF-REPLY-IDENTIFIER.

б) Преобразовать все ссылки на открытые типы к типам ANY или ANY EFINE BY. Для этого преобразовать весь синтаксис АСН.1 следующим образом:

во-первых, преобразовать

из	в
REGISTRE-AS.&id	BJECTIDENTIFIER

Если «REGISTERE-AS.&Type» является компонентом SET или SEQUENCE и определен в той же самой конструкции SET или SEQUENCE как «id», то преобразовать

из	в
REGISTERE-AS.&Type({RegisteredAsTable}{@.id1})	ANY EFINE BY id
REGISTERE-AS.&Type({RegisteredAsTable}{@.id1})	ANY EFINE BY id

в противном случае преобразовать

из	в
REGISTERE-AS.&Type({RegisteredAsTable}{@.id1})	ANY
REGISTERE-AS.&Type({RegisteredAsTable}{@.id1})	ANY

в) Если для модуля АСН.1 действует AUTOMATIC TAGS, то применить ИСО/МЭК 8824-1, пп. 22.5—22.7, где описано, как действует автоматическое тегирование на компоненты типов SET, SEQUENCE и CHICE, и удалить конструкцию «AUTOMATIC TAGS» из предложения определения модуля.

г) Если открытый тип ограничен с использованием нотации подтипа TypeConstraint, то удалить ограничение, так как ANY и ANY EFINE BY не могут быть ограничены в АСН.1:1990 до типа, отличного от ANY или ANY EFINE BY.

д) Если определение типа ENUMERATE использует синтаксис «identifier» для EnumerationItem, то его следует изменить на «identifier(number)». Например, изменить

ENUMERATE {a, b, c, d}

на

ENUMERATE {a (0), b (1), c (2), d (3)}

е) Удалить все маркеры расширения (т.е. «...»). Например, изменить

```
SEQUENCE{
    i    IA5String,
    b    B LEAN,
    ...
}
```

на

```
SEQUENCE{
    i    IA5String,
    b    B LEAN
}
```

ж) Удалить из предложения определения модуля все конструкции EXYENSIBILITYIMPLIE.

и) Удалить все символы пробелов и новой строки из hstring и bstring; удалить все символы новой строки из cstring. Например, изменить

b BITSTRING : : = '00011100110101101110011111010101'B

o CTET STRING : : = '8F3CE4830192B3459325EF28AA3E700'H

**p PrintableString : : = «Hello,
world»**

на

b BITSTRING : : = '00011100110101101110011111010101'B

o CTET STRING : : = '8F3CE4830192B3459325EF28AA3E700'H

p PrintableString : : = «Hello, world»

к) Преобразовать все нотации CharacterStringList в эквивалентные и мсstring. Например, изменить

**name PrintableString : : = {«This is a long string, that is
spread across two lines»}**

на

**name PrintableString : : =
«This is a long string, that is spread across two lines»**

л) Преобразовать все ссылки на множества значений в ссылки на ограниченные типы. Например, изменить

AgesINTEGER ::= { 1 | 4 | 7 .. 20 }

на

AgesINTEGER ::= { 1 | 4 | 7 .. 20 }

Примечание 2—Предпочтительнее использовать ограниченные типы вместо множеств значений, если только не используются интенсивно параметризация и классы информационных объектов, для которых полезно использование множеств значений.

м) Преобразовать все появления типа **INSTANCE F** в эквивалентный тип **SEQUENCE**. Например, изменить

A ::= **INSTANCE F REGISTERE -AS**

на

A ::= **SEQUENCE**{
 type-id **BJECT IDENTIFIER**,
 value **[0] ANY EFINE BY type-id**
 }

н) Удалить все идентификаторы «**mantissa**», «**base**» и «**exponent**» из всех значений типа **REAL** и заменить все внутренние ограничения типа **REAL** на комментарий. Например, изменить

ten REAL ::= { **mantissa 1, base 10, exponent 1** }
ecimalReal ::= **REAL (WITH COMPONENTS { ..., base 10 })**

на

ten REAL ::= { 1, 10, 1 }
ecimalReal ::= **REAL**--должно кодироваться по основанию 10

о) Заменить все случаи нотации значения **EXTERNAL** на эквивалентные ей в АСН.1:1990. Например, изменить

extern1990EXTERNAL ::= {
 direct-reference { 1 2 3 4 5 6 },
 indirect-reference 3,
 encoding **single-ASN1-type:IA5String:«hello»**
 }

на

extern1994EXTERNAL ::= {
 identification-context-negotiation: {
 presentation-context-id 3,
 transfer-syntax { 1 2 3 4 5 6 }
 },
 data-value notation: **IA5String:«hello»**
 }

п) Заменить все допустимые алфавиты АСН.1:1994 на эквивалентные им в АСН.1:1990. Например, изменить

UpperCaseAndSpacently ::= **PrintableString (FRM("A".."Z"|" "))**

и

UpperCaseAndSpacently ::= **PrintableString (FRM("A"|"B"|"C"|" "|
 "E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|"N"|" "|
 "P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"))**

р) Изменить все выражения множеств АСН.1:1994, используемые в нотации подтипа, на эквивалентные им в АСН.1:1990. Например, изменить

PartNumber ::= **NumericString(SIZE(8)^FRM("0".."9"))**

на

PartNumber ::= **NumericString(SIZE(8))(FRM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"))**

с) Когда требование р) не может быть выполнено, так как результатом будет бесконечное множество, часть нотации подтипа АСН.1:1994, которая приводит к бесконечному множеству, следует заменить комментарием. Например, изменить

```
AllButZeroToTen : : = INTEGER(ALLEXCEPT(0..10))
```

на

```
AllButZeroToTen : : = INTEGER -- все целые значения, кроме 0 — 10
```

Применение приведенных выше инструкций к модулю АСН.1:1994 **ExampleModule** из перечисления 5) дает:

```
-- < Модуль АСН.1 версии 1990 года ExampleModule - -
```

```
ExampleModule{-- здесь должен быть допустимый идентификатор объекта-
```

```
EFINITINS : : = BEGIN
```

```
Foo : : = SEQUENCE{
```

```
  id1 BJESENTIFIER,
```

```
  syntax1 ANY EFINE BY id1}
```

```
Bar : : = SEQUENCE{
```

```
  id2 BJESENTIFIER,
```

```
  syntax2 ANY}
```

```
-- В модуле АСН.1:1994 ExampleModule синтаксис syntax2 в Bar был определен как
```

```
-- SEQUENCE F, а не как открытый тип, и, таким образом, не мог быть преобразован в
```

```
-- ANY EFINE BY.
```

```
firstExtensionId BJESENTIFIER : : = {1 3 17 103 101}
```

```
FirstExtensionInfo : : = PrintableString
```

```
-- firstExtensionId и FirstExtensionInfo должны использоваться в типе Foo, где firstExtensionId
```

```
-- является значением mid1 (BJESENTIFIER), которое указывает, что syntax1 имеет тип
```

```
-- синтаксиса FirstExtensionInfo(PrintableString).
```

```
EN
```

10 Соглашения для АСН.1 и директив РОУО

В настоящем разделе вводятся соглашения для ясной идентификации спецификации и других пользовательских возможностей, связанных с шаблонами РОУО, и относящихся к ним модулей АСН.1. Это осуществляется с помощью директив в потоке. Настоящие соглашения могут быть полезны в качестве директив для компиляторов как АСН.1, так и РОУО. Если используются настоящие соглашения, то не требуется, чтобы автор изменял спецификации АСН.1 и РОУО для использования этих директив; а именно, директивы могут находиться в том же тексте, что и модули АСН.1 или шаблоны РОУО, но могут находиться и в других спецификациях. Если используются настоящие соглашения, то они не изменяют спецификаций АСН.1 или РОУО; а именно, эти директивы не изменяют синтаксис спецификаций АСН.1 или РОУО. Настоящие соглашения являются рекомендуемыми, но не обязательными.

Предлагаемые директивы построены так, чтобы удовлетворить следующим требованиям:

- входные файлы директивами (т.е. модулями АСН.1, библиотеками РОУО и прочими директивами) должны быть приняты компиляторами АСН.1 и РОУО, которые не распознают директив;
- входные файлы без директив должны быть приняты компиляторами АСН.1 и РОУО.

Примечание — Эти соглашения допускают расширения путем использования специфичных для реализации директив.

Каждая спецификация директивы является структурированным комментарием, содержащим единственную директиву, которая состоит из ключевого слова, квалифицированного областью действия, с последующими нулем или несколькими операндами. Регистр принимается во внимание. Следовательно, директивы рассматриваются как комментарии либыми компиляторами, которые этих директив не поддерживают.

Для описания директив использованы следующие соглашения:

- текст, набранный жирным шрифтом (например, --< и ли **ASN1**), должен вводиться так, как он приведен, без добавления пробелов или изменения регистра;
- текст, набранный курсивом (например, *ключевое_слово*), должен быть заменен подходящим текстом;
- факультативные элементы директив заключены в квадратные скобки (например, [операнды]);

- за элементом директивы, который может повторяться (произвольное число раз), стоят три точки (например, [операнды]...).

Общий формат директивы:

--<директива--

где директива есть

область_действия.ключевое_слово[операнд][,операнд]...

Таким образом, директива начинается двумя последовательными дефисами и знаком «меньше, чем» (--<) и завершается знаком «больше, чем» и двумя последовательными дефисами (--). Между дефисами и знаками «меньше, чем» и «больше, чем» не должно быть пробелов. В результате компиляторами, которые не поддерживают эти директивы, они трактуются как комментарии АСН.1 или РОУО. Директива не может содержать комментарий АСН.1.

Каждая конструкция --<-- является структурированным комментарием, содержащим единственную директиву, которая состоит из ключевого слова, квалифицированного областью действия, за следующими нулевыми или несколькими операндами. Региструется.

Директива может продолжаться на следующих строках, первыми отличными от пробела символами которых должны быть два дефиса (--).

Операнды разделяются одним или несколькими последовательными пробелами или символами табуляции — для несхожих элементов, или запятой — для списка схожих элементов (таких как имена рабочих наборов). Пробелы или символы табуляции могут находиться до и после других элементов директивы. В данном контексте символы возврата каретки, новой строки и вертикальной табуляции не рассматриваются как пробелы и являются недопустимыми.

10.1 Соглашения для директив АСН.1

Для директив АСН.1 приняты следующие соглашения.

Директива может находиться в том же файле, что и элемент АСН.1. В таком случае директива может располагаться вне области действия модуля АСН.1 (до его начала или после его конца). Директива может находиться в теле модуля, в любом месте, где допустим пробел. В этом случае каждая директива должна помещаться до того элемента АСН.1, к которому она применяется.

Для директивы АСН.1 символом область-действия является ASN1. В дополнении к нему могут быть определены (например, реализацией) другие символы область-действия.

Ключевым-словом может быть следующее:

- **Version.**

Операндом, в общем случае, могут быть:

- текстовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);

- числовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);

- cstring («xxxxx»), занимающая одну строку;

- {идентификатор-объекта};

- другие конструкции АСН.1, например bstring или hstring.

- текстовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);

- числовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);

- cstring («xxxxx»), занимающая одну строку;

- {идентификатор-объекта};

- другие конструкции АСН.1, например bstring или hstring.

10.1.1 Директива версии

Директива **Version** используется для указания того, по какой версии написан модуль АСН.1: АСН.1:1990 или АСН.1:1994.

Директива имеет следующий формат:

--<ASN1. **Version** версия имя_модуля [фио] --

Элементы, выделенные жирным шрифтом (например, **ASN1. Version**) пишутся так, как показано, а элементы, набранные курсивом, заменяются следующим образом:

версия — либо **1990**, либо **1994**, либо **1990, 1994**;

имя_модуля — имя модуля АСН.1;

фио — факультативное значение идентификатора объекта АСН.1, используемое для недвусмысленной идентификации модуля АСН.1.

Директива **Version** является единственной директивой для модуля АСН.1. Эта директива со значением операнда 1990, 1994 означает, что модуль АСН.1 соответствует как версии АСН.1:1990, так и версии АСН.1:1994. Компилятор может использовать свои сведения о любой из этих версий или об их общем подмножестве. Если директива версии для модуля АСН.1 не предоставлена, то реализация может использовать какой-либо иной способ для идентификации версии модуля, например, опции командной строки компилятора или распознавание синтаксиса, специфичного для конкретной версии (как макров АСН.1:1990 или информационных объектов АСН.1:1994).

Примеры:

```
-- <ASN1.Version 1990 Attribute-ASN1Module
-- {joint-iso-itu-tms(9) smi(3) part2(2) asn1Module(2) 1} --
-- <ASN1.Version 1994 SMMModule
-- {joint-iso-itu-tms(9) smi(3) part4(4) asn1Module(2) 2} --
```

10.2 Соглашения для директив РОУО

Для директив РОУО существуют следующие дополнительные соглашения.

Директива не может содержать комментарий в АСН.1.

Директива может находиться как в файле, отличном от содержащего шаблон РОУО, к которому она относится, так и в самом файле. Когда директива находится в том же файле, она может быть вне области действия документа РОУО (до начала или после его). Директива может находиться в теле документа, в любом месте, где допустим пробел.

Когда директива находится внутри документа, она должна располагаться до шаблона РОУО, к которому она относится. Для конкретного шаблона РОУО может существовать не более одной директивы конкретного вида, но могут существовать разные директивы для одного и того же шаблона. Например, на один и тот же шаблон могут ссылаться директивы **Nickname** и **WorkingSet**, но только по одной каждого вида. Для других элементов АСН.1 могут существовать другие директивы **Nickname**.

Символом области действия является G MO. Дополнительно могут быть определены (например, реализацией) другие символы области действия.

Ключевым словом может быть одно из следующих:

- **Alias**;
- **ocument**;
- **Endocument**;
- **Version**.

Операндом, в общем случае, могут быть:

- текстовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);
- числовая строка (без пробелов, запятых, последовательной пары дефисов и знака «больше, чем»);
- cstring («xxxxx»), занимающая одну строку;
- {идентификатор-объекта};
- другие конструкции АСН.1, например pbstring или hstring.

10.2.1 Директива альтернатив

Директива **Alias** используется для предоставления альтернативных или алиасных идентификаторов документа РОУО. Эти алиасы используются для согласования ссылок между документами РОУО, когда используются короткие или противоречивые идентификаторы.

Формат директивы:

```
-- < G M . Alias идентификатор_документа алиас_документа
-- [, алиас_документа] ... --
```

Элементы, выделенные жирным шрифтом (например **G M . Alias**) пишутся так, как показано, а элементы, набранные курсивом (например *идентификатор_документа*), заменяются так, как описано ниже.

Может присутствовать несколько разделенных запятыми элементов алиас_документа.

- идентификатор_документа — идентификатор документа РОУО в виде либо строки, взятой в кавычки ("), либо идентификатора объекта в фигурных скобках ({});
- алиас_документа — альтернативный идентификатор документа РОУО в виде либо строки, взятой в кавычки ("), либо идентификатора объекта в фигурных скобках ({}).

Примеры:

```
--< G M . Alias "CCITT Rec. X.721 (1992)|IS/IEC 10165-2:1992"
-- "Rec. X.721|IS/IEC 10165-2:1992",
-- "CCITT Rec. X.721|IS/IEC 10165-2",
-- "Rec. X.721|IS/IEC 10165-2",
-- "CCITT Rec. X.721 (1992)",
-- "CCITT Rec. X.721 (1992)|IS/IEC 10165-2:1992",
-- " M I " --
--< G M . Alias "Recomendation M.3100:1992"
-- "Rec.M.3100:1992",
-- "M.3100:1992",
-- "M. 3100" --
```

10.2.2 Д и р е к т и в а д о к у м е н т а

Директива `osument` обеспечивает идентификатор документа РОУО, являющийся либо символьной строкой, либо идентификатором объекта, либо тем и другим. Формат директивы может иметь одну из следующих трех форм:

```
--< G M . o s u m e n t строка_документа --
--< G M . o s u m e n t ИО-документа --
--< G M . o s u m e n t строка_документа ИО-документа --
```

Элементы, выделенные жирным шрифтом (например **G M . o s u m e n t**) пишутся так, как показано, а элементы, набранные курсивом (например *строка_документа*), заменяются следующим образом:

строка_документа—символьная строка, идентифицирующая документ РОУО, в кавычках ("");
ИО-документа—идентификатор объекта АСН.1, присвоенный документу, в фигурных скобках ({}).

Директива `osument` должна находиться в допервого шаблона РОУО или модуля АСН.1, образующего документ. Таким образом, документ РОУО рассматривается как состоящий из всех шаблонов РОУО и модулей АСН.1 от директивы `osument` до соответствующей директивы `Endocument`, или до следующей директивы `osument`, или до конца файла, содержащего этот текст РОУО.

Для сохранения в файлах текстов РОУО, имеющих соответствующие директивы, существуют следующие правила:

- Каждый документ РОУО должен иметь директиву `osument` для обеспечения «правильного» имени документа. Если этой директивы нет, то имя документа не определено или обеспечивается каким-либо другим методом.

- Директива `osument` должна находиться в том же самом файле, что и текст РОУО, к которому она применяется.

Одно и то же имя документа РОУО может появиться в нескольких директивах `osument` применительно к нескольким блокам текста РОУО, таким как различные файлы. В этом случае весь текст РОУО в разных файлах рассматривается как часть одного и того же документа РОУО. Это аналогично пространству имен в C++, которое позволяет в нескольких разных файлах заголовков содержать материал, определенный в одном и том же пространстве имен.

В общем случае должен быть единственный документ РОУО на один входной файл. Файл может содержать несколько документов РОУО только в том случае, когда он содержит несколько согласованных директив `osument` и `Endocument`.

Пропуски (одни или несколько последовательных пробелов и дисимволов табуляции) не учитываются в идентификаторе `строка_документа` и `валиасе_документа`.

Примеры:

```
--< G M . o s u m e n t "CCITT Rec. X.721 (1992)|IS/IEC 10165-2:1992" --
--< G M . o s u m e n t "Recomendation M.3100:1992" --
--< G M . o s u m e n t " P 1 L i b r a r y V o l . 4 " --
--< G M . o s u m e n t { i s o ( 1 ) 2 1 2 4 3 6 0 5 0 1 1 5 1 3 1 } --
--< G M . o s u m e n t " I I M C M I B T r a n s l a t i o n " { I S ( 1 ) 2 1 2 4 3 6 0 5 0 1 1 5 1 3 1 } --
```

10.2.3 Д и р е к т и в а к о н ц а д о к у м е н т а

Директива `Endocument` отмечает конец "документа" РОУО. Формат директивы может иметь одну из следующих четырех форм:

```
--<G M .E n d o c u m e n t - -
--<G M .E n d o c u m e n t строка_ документа --
--<G M .E n d o c u m e n t ИО- документа --
--<G M .E n d o c u m e n t строка_ документа ИО- документа --
```

Элементы, выделенные жирным шрифтом (например **G M .E n d o c u m e n t**), пишутся так, как показано, а элементы, набранные курсивом (например *строка_ документа*), заменяются следующим образом:

строка_ документа — символьная строка, идентифицирующая документ РОУО, в кавычках ("");
ИО- документа — идентификатор объекта АСН.1, присвоенный документу, в фигурных скобках ({}).

Документ РОУО рассматривается как состоящий из всех шаблонов РОУО и модулей АСН.1 от директивы *ocument* до соответствующей директивы *Endocument*, или до следующей директивы *ocument*, или до конца файла, содержащего этот текст РОУО. Таким образом, использование директивы *Endocument* не обязательно. Если же директива *Endocument* используется, то она должна следовать за последним шаблоном РОУО или модулем АСН.1, образующем документ. Строка_ документа и (или) ИО- документа в директиве **Endocument** должна(ы) соответствовать предшествующей директиве **ocument**.

П р и м е р ы

```
--<G M .E n d o c u m e n t " - -
--<G M .E n d o c u m e n t " C C I T T R e c . X . 7 2 1 ( 1 9 9 2 ) | I S / I E S 1 0 1 6 5 - 2 : 1 9 9 2 " - -
--<G M .E n d o c u m e n t " R e c o m e n d a t i o n M . 3 1 0 0 : 1 9 9 2 " - -
--<G M .E n d o c u m e n t " P 1 L i b r a r y V o l . 4 " - -
--<G M .E n d o c u m e n t { i s o ( 1 ) 2 1 2 4 3 6 0 5 0 1 1 5 1 3 1 } - -
--<G M .E n d o c u m e n t " I I M C M I B T r a n s l a t i o n " { i s o ( 1 ) 2 1 2 4 3 6 0 5 0 1 1 5 1 3 1 } - -
```

10.2.4 Д и р е к т и в а в е р с и и

Директива **Version** используется для указания того, какая версия РОУО использована в документе. Формат директивы может иметь одну из следующих двух форм:

```
--<G M .V e r s i o n версия - -
--<G M .V e r s i o n версия идентификатор_ документа --
```

Элементы, выделенные жирным шрифтом (например **G M .V e r s i o n**), пишутся так, как показано, а элементы, набранные курсивом (например *версия*), заменяются следующим образом:

- версия — номер, указывающий версию РОУО, в соответствии со следующей таблицей:

Индекс	Определение версии
1	РОУО, 1992
1.1	Дополнение 1: конструкция SET-BY-CREATE
1.2	Дополнение 2: конструкции N - M I F I , A S N . 1 : 1 9 9 4 и директивы
1.3	Дополнение 3: использование Z в поведении класса управляемых объектов

Новые версии определяются при каждом обновлении РОУО (т. е., пересмотре, поправках, дополнениях). Обеспечение данной версии является кумулятивным. Это значит, что любой документ РОУО, действительный для версии *n*, должен быть допустимым и для предшествующих версий. Например **G M .V e r s i o n 1 . 2** указывает, документ РОУО поддерживает версию РОУО 1992 (версия 1), дополнение конструкции **SET-BY-CREATE** (версия 1.1) и дополнение конструкций **NO-MOIFI, ASN.1:1994** и директив (версия 1.2).

- идентификатор_ документа — идентификатор документа РОУО в виде символьной строки в кавычках ("") или идентификатора объекта АСН.1 в фигурных скобках ({}).

Первая форма директивы (без идентификатора_ документа) может находиться в документе РОУО, но любого шаблона РОУО.

П р и м е р ы

```
--<G M .V e r s i o n 1 " C C I T T R e c . X . 7 2 1 ( 1 9 9 2 ) | I S / I E C 1 0 1 6 5 - 2 : 1 9 9 2 " - -
--<G M .V e r s i o n 1 . 1 " R e c o m e n d a t i o n M . 3 1 0 0 " - -
--<G M .V e r s i o n 1 . 2 - -
```

ПРИЛОЖЕНИЕ А
(справочное)

Примеры

Примеры, приведенные в настоящем приложении, иллюстрируют использование шаблонов, а не содержат полезной для реализации информации. В частности, определения поведения являются искусственными. Примеры, имеющие практическое значение для разработки определений классов управляемых объектов, приведены в ГОСТРИСО/МЭК10165-2.

А.1 Определение класса управляемых объектов

```
example bjectClass MANAGE BJECT CLASS
  ERIVE FRM "CCITT Rec. X.721 (1992) | IS/IEC 10165-2:1992": top
  CHARACTERIZE BY examplePackage2;
  CNITINAL PACKAGES
    examplePackage1 PACKAGE
      ACTI NS      q SResetAction,
                  activate;
      NTIFICATINS communicationError;
  REGISTERE AS {joint-iso-ccittms(9) smi(3) part4(4) package(4) examplepack1(0)};
  PRESENTIF |применяется класс соответствия 2 ниже лежащего ресурса,
            описанный в ИСО/МЭК XXXX|;
REGISTERE AS {joint-iso-ccittms(9) smi(3) part4(4)
  managed bjectClass (3) exampleclass (0)};
```

Примечание — Этот шаблон использует возможность встроенного документирования условного пакета.

А.2 Определение связывания имен

```
example NameBinding NAMEBINING
  SUBRINATE BJECT CLASS example bjectClass;
  NAME BY
  SUPERIR BJECT CLASS "CCITT Rec. X.721 (1992) | IS/IEC 10165-2:1992": system;
  WITH ATTRIBUTE objectName;
  BEHAVIUR
    containmentBehaviour BEHAVIUR
      EFINE AS | Влюбом экземпляре "CCITT Rec. X.721 (1992)
                | IS/IEC 10165-2:1992": system может содержаться
                не более трех экземпляров example bjectClass |
      ;
    ;
  CREATE WITH-AUTOMATIC-INSTANCE-NAMING createErrorParameter;
  ELETE ELETES - CNTAINE - BJECTS;
REGISTERE AS {joint-iso-ccittms(9) smi(3) part4(4) nameBinding (6) examplenb (0)};
```

Примечание — Этот шаблон использует возможность встроенного документирования поведения.

А.3 Определение параметров

```
pUHeader PARAMETER
  CNTXT      EVENT-INF;
  WITHSYNTAX ParameterModule.PUString;
  BEHAVIUR
    pUHeaderBehaviour BEHAVIUR
      EFINE AS |Заголовок ПБД.
                Передается в поле ПОИУ eventInfo. |
      ;
    ;
REGISTERE AS {joint-iso-ccittms(9) smi(3) part4(4) parameter(5) pduheaderparam(0)};
createErrorParameter PARAMETER
  CNTXT      SPECIFIC-ERRR;
  WITHSYNTAX ParameterModule.ErrorInfo1;
  BEHAVIUR
```

createErrorBehaviour BEHAVIUR

E F I N E A S |Если в объекте управляемого объекта существует максимальное возможное число экземпляров **exampleObjectClass**, то попытка создания дополнительных экземпляров приведет к возврату сообщения об ошибке ПОИУ «Отказ обработки», в котором поле **SpecificErrorInfo** имеет вид
SpecificErrorInfo ::= SEQUENCE {
errorid BJECTIDENTIFIER,
errorinfo ANY E F I N E B Y **errorid** }
BJECTIDENTIFIER, передаваемый в **errorid**, должен быть значением параметра, под которым оно зарегистрировано. Тип, передаваемый в **errorinfo**, должен быть идентифицированным конструкцией **WITHSYNTAX** этого определения параметра. Значение, передаваемое типом, указывает число экземпляров этого класса управляемых объектов, которые в данный момент существуют в объекте.

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) parameter (5) createrror (1)};

serviceProviderErrorResponseReason PARAMETER

C N T E X T ACTIN-REPLY;

WITHSYNTAX ParameterModule.ServiceProviderErrorResponseReason;

BEHAVIUR

serviceProviderErrorResponseReasonBehaviour BEHAVIUR

E F I N E A S |Возвращается в поле **responsePARAMETERS** ПОИУ **actionReplyInfo**, если **responseCode** имеет текущее значение **serviceProviderErrorResponse**.

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) parameter (5) sperrorrsp (2)};

Примечание — Этот шаблон использует возможность встроенного документирования поведения.

А.4 Определение пакета

examplePackage2 PACKAGE

BEHAVIUR exampleClassBehaviour;

ATTRIBUTES objectName

GET,

qS-Error-Cause

GET,

qS-Error-Counter

PERMITTERVALUESAttributeModule.QSCounterRange

REQUIREVALUESAttributeModule.QSCounterRange

GET;

ATTRIBUTEGRUPS qS-Group;

NTIFICATINS protocolError;

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) package (4) examplepack2 (1)};

Примечание — Так как этот шаблон не используется в качестве условного пакета, то конструкция **REGISTEREAS** не является строго обязательной, но просея включить регистрацию во время спецификации, чем добавлять ее позже, если она станет необходимой для использования этого пакета в качестве условного.

А.5 Определения атрибутов

objectName ATTRIBUTE

WITHATTRIBUTESYNTAXAttributeModule.bjectName;

MATCHESFR EQUALITY;

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) attribute (7) objectname (0)};

qS-Error-CauseATTRIBUTE

WITHATTRIBUTESYNTAXAttributeModule.QSErrorCause;

MATCHESFR EQUALITY;

BEHAVIUR qSErrorBehaviour;

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) attribute (7) qoscause (1)};

qS-Error-CounterATTRIBUTE

WITHATTRIBUTESYNTAXAttributeModule.QSErrorCounter;

MATCHESFR EQUALITY, RERING;

BEHAVIUR qSCounterBehaviour;

REGISTEREAS {joint-iso-ccittms (9) smi (3) part4 (4) attribute (7) qoscount (2)};

А.6 Определение атрибутивной группы

q S-Group ATTRIBUTE GRUP

GRUPELEMENTS q S-Error-Cause, q S-Error-Counter;

E S C R I P T I N |Атрибутивная группа, которая включает все атрибуты, относящиеся к качеству услуг в классе управляемых объектов.;

REGISTEREAS {joint-iso-ccittms(9) smi(3) part4(4) attributeGroup(8) qosgroup(0)};

А.7 Определения действий

q S ResetAction ACTIN

BEHAVIUR

reset BEHAVIUR

E F I N E A S |<Определение поведения reset и его влияния на работу управляемого объекта и т. п. |

;

;

M E C N F I R M E ;

REGISTEREAS {joint-iso-ccittms(9) smi(3) part4(4) action(9) reset(0)};

Примечание—Это определение действия использует возможность встроенного документирования поведения. Абстрактные синтаксисы для вызова и ответа не определены.

activate ACTIN

BEHAVIUR

activateBehaviour BEHAVIUR

E F I N E A S | Делает управляемый объект доступным для работы. Если действие успешное, то возвращается значение successResponse в параметре responseCode ПО ИУ actionReplyInfo. Если действие неудачное из-за проблем поставщиком ниже следующих услуг, то responseCode устанавливается равным serviceProviderErrorResponse и возвращается параметр serviceProviderErrorResponseReason для указания причины проблемы. |

;

;

M E C N F I R M E ;

PARAMETERS serviceProviderErrorResponseReason;

WITHREPLYSYNTAX ActionModule.ActivateReply;

REGISTEREAS {joint-iso-ccittms(9) smi(3) part4(4) action(9) activate(1)};

А.8 Определения сообщений

communicationError NOTIFICATION

BEHAVIUR

communicationErrorBehaviour;

WITHINFORMATINSYNTAX NotificationModule.ErrorInfo;

WITHREPLYSYNTAX NotificationModule.ErrorResult;

REGISTEREAS {joint-iso-ccittms(9) smi(3) part4(4) notification(10) commerror(0)};

protocolError NOTIFICATION

BEHAVIUR

protocolErrorBehaviour

BEHAVIUR

E F I N E A S | Создается, когда протокольный объект получает ПБД, который является недопустимым или содержит ошибку протокола. Сообщение включает все заголовки полученного ПБД. |

;

;

PARAMETERS p U Header;

WITHINFORMATINSYNTAX NotificationModule.ProtocolError;

REGISTEREAS {joint-iso-ccittms(9) smi(3) part4(4) notification(10) protoerror(1)};

Примечание—Этот шаблон использует возможность встроенного документирования поведения.

А.9 Определения поведения

q S CounterBehaviour BEHAVIUR

E F I N E A S | Атрибут q S ErrorCounter является счетчиком, который увеличивается на единицу при каждом появлении ошибки КУ. Его значения является положительным целым, диапазон которого специфицируется в пакетах, ссылающихся на это определение. Когда счетчик достигает максимального значения, следующее увеличение вызывает возврат значения нулю. |;

qSErrorBehaviour BEHAVIUR
EFINE AS | Атрибут **qSErrorCause** указывает причину отказа КУ, связанной с управляемым объектом.

Примечание— Взаимоотношения между допустимыми значениями атрибута и работой самого управляемого объекта определяются определениями поведения, связанного с определением класса управляемых объектов.;

communicationErrorBehaviour BEHAVIUR
EFINE AS | Сообщение **CommunicationError** создается классом управляемых объектов, когда управляемый объект обнаруживает ошибку коммуникации. Сообщение может содержать любую комбинацию параметров **ProbableCause**, **Severity**, **TrendIndication**, **BackedUpStatus**, **diagnosticInfo**, **ProposedRepairAction**, **ThresholdInfo**, **StateChange** и **therInfo**.

Примечание— Точное определение того, что представляет собой ошибка коммуникации и используемых значений параметров, специфично для классов управляемых объектов. На практике это определение поведения может ссылаться на базовые стандарты.;

exampleClassBehaviour BEHAVIUR
EFINE AS | <...Описание поведения класса управляемых объектов, включая описания:
 - как атрибуты получают конкретные значения и какой они имеют смысл;
 - какие обстоятельства вызывают создание сообщений;
 - и т. п. . ;

A.10 Модули ASN.1

AttributeModule{joint-iso-ccittms(9)smi(3)part4(4)asn1Module(2)attributes(0)};

EFINITI NS ::= BEGIN

bjectName ::= GraphicString

QSErrorCause ::= INTEGER {
 responceTimeExcessive (0),
 queueSizeExceeded (1),
 bandwidthReduced (2),
 retransmissionRateExcessive (3) }

QSErrorCounter ::= INTEGER

QSCounterRange ::= QSErrorCounter {0..4294967296}— Диапазон 32 бита.

EN

NotificationModule{joint-iso-ccittms(9)smi(3)part4(4)asn1Module(2)notifications(1)};

EFINITI NS ::= BEGIN

IMP RTS

ProbableCause, **PerceivedSeverity**, **TrendIndication**, **BackedUpStatus**, **ProposedRepairActions**, **ThresholdInfo**, **ManagementExtension**

FRMAttribute.ASN1Module{joint-iso-ccittms(9)smi(3)part2(2)asn1Module(2)1};

ErrorInfo ::= SET {
 [0]**ProbableCause** PTI NAL,
 [1]**PerceivedSeverity** PTI NAL,
 [2]**TrendIndication** PTI NAL,
 [3]**BackedUpStatus** PTI NAL,
 [4]**ProposedRepairActions** PTI NAL,
 [5]**ThresholdInfo** PTI NAL,
 [6]**therInfo** PTI NAL }

ErrorResult ::= NULL

therInfo ::= SET FManagementExtension

ProtocolError ::= SET FManagementExtension

EN

ActionModule{joint-iso-ccittms(9)smi(3)part4(4)asn1Module(2)actions(2)}

EFINITI NS ::= BEGIN

IMP RTS

perationalState, **ManagementExtension**

FRMAttribute.ASN1Module{joint-iso-ccittms(9)smi(3)part2(2)asn1Module(2)1};

ActivateReply ::= SEQUENCE {
 operationalStatus [0]**perationalState**,
 responseCode [1]INTEGER{**successResponse** (0),
 serviceProviderErrorResponse (1)},
 responseParams [2] SET FManagementExtension PTI NAL }

EN

```

ParameterModule{joint-iso-ccittms(9)smi(3)part4(4)asn1Module(2)parameters(3)}
E FINIT I NS : : = BEGIN
ErrorInfo1 : : = INTEGER
ServiceProviderErrorResponseReason : : = ENUMERATE {
    insufficientResources      (0),
    provideroesNotExist        (1),
    providerNotAvailable        (2),
    requiredServiceNotAvailable (3) }
P UString : : = СТЕТSTRING
E N

```

ПРИЛОЖЕНИЕ В (справочное)

Руководство по применению Z при формализации поведения управляемых объектов

В.1 Введение

Настоящее приложение содержит техническое руководство по применению языка Z для определения поведения управляемых объектов, которые поддерживают административное управление взаимодействием ВОС. Оно является справочным, а не нормативным. Оно не требует, чтобы для спецификации поведения УО использовались методы формального определения (МФО). Если требуется, чтобы использовались МФО, то тем самым не требуется, чтобы использовался Z; могут использоваться другие языки, такие как SL. Даже когда должен использоваться Z, возможны и другие способы спецификации поведения УО.

Формальные спецификации поведения УО могут быть непосредственно полезными, так как они ясны и недвусмысленны. Процесс создания формальной спецификации заставляет проводить подробный анализ поведения. Следовательно, формальная спецификация может использоваться как инструмент для идентификации и исправления двусмысленностей, которые могли остаться невыявленными в спецификации, полностью основанной на естественном языке. По этим причинам формальная спецификация может быть полезной для совершенствования спецификации поведения.

Настоящее приложение содержит иллюстрированный пример, который показывает современную практику использования Z. Его целью является установление тех общих основ и понимания этого конкретного формального подхода, которые помогут достичь согласованности в сходных разработках. Он предоставляет полезную начальную точку для пользователей РОУО, которые хотят использовать Z для улучшения своих спецификаций.

Приложение предназначено для пользователей, знакомых с основными понятиями спецификаций управляемых объектов, использующих шаблоны РОУО, и с языком Z.

В настоящем приложении термин «управляемый объект» (или «УО») используется для определения класса управляемых объектов, данных с использованием шаблонов РОУО.

В.2 Вопросы языка

Нотация Z является формально определенной нотацией, основанной на теории множеств и исчисления предикатов. Она имеет достаточно мощные средства для описания отдельных классов управляемых объектов.

Однако в Z не существует понятие инкапсуляции. Спецификация Z обычно состоит из модели некоторого состояния и совокупности операций для изменения этого состояния. В Z нет методов для деления состояния и его операций на отдельные модули и их повторного использования в других спецификациях. Очевидным следствием этого является необходимость описания управляемых объектов, которые наследуют переменные и поведение из других определений классов управляемых объектов.

Эффект наследования может быть получен методом включения схемы с некоторой потерей ясности. Вовсе остальных отношениях Z подходит для выражения отдельных классов управляемых объектов.

В.3 Элементы, подлежащие преобразованию

Требуется преобразовать определение поведения (или его часть) из неформального описания в описание Z. Степень, в которой должны быть формализованы оставшиеся части шаблонов РОУО, зависит, главным образом, от потребностей разработчика спецификации.

Шаблоны РОУО уже включают в себя полуформальные определения типов данных в АСН.1. Можно написать спецификацию Z, используя эти определения АСН.1 в качестве основы для типов, используемых в спецификации Z, что сэкономит существенную часть работы.

Однако если спецификация написана таким образом, то большой проблемой для разработчика становится гарантия ее синтаксической корректности. В спецификациях Z без определений АСН.1 можно исполь-

зовать существующие средства Z , которые обеспечивают поддержку проверки синтаксиса и статической семантики спецификации Z .

Таким образом, можно улучшить определения поведения, используя Z без переписывания типов данных АСН.1, но существенные преимущества могут быть получены при полном преобразовании типов данных АСН.1 в Z . Примеры того, как преобразовываются основные типы АСН.1 в Z , приведены в В.7.1.

В.3.1 Преобразование из шаблонов РОУО в Z

Ниже дано общее руководство по преобразованию управляемых объектов в их неформальное описание настоящего стандарта в Z . Такое преобразование может быть осуществлено только неформально, т.к. формальное преобразование требует, как минимум, чтобы исходный и целевой языки были формализованы. Более того, как и любое отображение двух различных языков, оно ограничено некоторым несоответствием между конструкциями этих языков. Эта проблема усугубляется, когда один из языков оказывается неформальным или включает в себя неформальные компоненты.

Ниже перечислены некоторые основные характеристики шаблонов, определенных в настоящем стандарте, с указанием их отличий от соответствующих конструкций Z . Попутно предлагаются общие методы разрешения несогласованности или рекомендации по их разрешению в конкретных случаях.

В настоящем приложении основное внимание уделяется тому, что необходимо описывать в поведении управляемого объекта. Дополнительная информация о преобразовании типов АСН.1 приведена в В.6.

В.3.2 Типы данных

Первым шагом является переписывание типов данных настоящего стандарта в виде типов Z . АСН.1 предоставляет полезные возможности по созданию типов данных, но их построение ориентировано на описание потоков данных, передаваемых между системами.

В АСН.1 построение типа определяется в виде списка. В Z типы являются множествами. Хотя можно моделировать построение типа АСН.1 в виде последовательности в Z , иногда бывает более естественным рассмотреть операции, доступные над типами АСН.1, и отобразить их в типы Z , чтобы более ясно описывалась структура. Типы АСН.1 «последовательность» и «множество» могут быть отображены в тип Z «кортеж». Тип АСН.1 «последовательность-из» может быть отображен в тип Z «последовательность». Тип АСН.1 «множество-из» может быть отображен в тип Z «множество».

АСН.1 включает в себя специальное обеспечение кодирования, такое как метки и значения по умолчанию. Оно не обязательно должно быть представлено в Z , т.к. не влияют на определение поведения.

В пункте В.6.2 приведена дополнительная информация о преобразовании типов АСН.1.

В.3.3 Атрибуты УО

Управляемые объекты определены таким образом, что имеют некоторые атрибуты административного управления. Эти атрибуты имеют типы данных, определенные в АСН.1. Им присвоены идентификаторы объектов. Они могут иметь свойства согласования значений. Предлагаются два способа моделирования таких атрибутов:

- простые типы атрибутов
- типы атрибутов как схемы.

Простейшим подходом является представление атрибута УО как переменной Z с соответствующим типом данных. Тогда отдельно необходимо определение постоянной, представляющей идентификатор объекта для этого атрибута. Эта постоянная будет связана с атрибутом только по соглашению. Когда для атрибута определена операция согласования, можно использовать свойство фактического фиксированного согласования. Пример приведен в В.6.3.

Можно включить все эти свойства атрибута в единственном типе схемы, который будет типом переменной Z , моделирующей атрибут УО. Схема будет включать в себя значение атрибута, идентификатор объекта и свойство согласования. Пример приведен в В.6.4. Когда требуются правила согласования, отличные от равенства, можно определить параметр согласования как отношение Z над типом значения атрибута. Этот подход допускает формальное представление произвольных конкретных правил согласования, что может быть важным для области действия, фильтрации и выбора объекта.

Трудно моделировать тип ANY АСН.1 в Z . В некоторых случаях можно дать список значений атрибута. Таким образом полная формальная модель, вероятно, потребует комбинации свободного типа Z с уже определенными типами атрибутов. Примеры приведены в В.6.1 и В.6.5.

Идентификаторы объектов формально моделируются множеством:

[OBJECT]

В.3.4 Другие идентификаторы объектов

Многие элементы, кроме атрибутов, имеют идентификаторы объектов. Удобно вести их все как постоянные в аксиоматические определения. Можно использовать соглашение о дополнении их суффиксом «Oid». Такие постоянные будут необходимы для классов, пакетов и сообщений.

Пример

```
packagesPackageOid : OBJECT
allomorphsPackageOid : OBJECT
topClassOid : OBJECT
```


В.3.5 Наследование и совместимость

Z может быть использован для построения иерархий наследования УО путем включения схем в модель наследования и специализации. Таким образом корректно моделируется поведение классов и подтипов УО, но это не позволяет явно выразить взаимоотношение строго подтипа, которое реально существует. Необходим язык, который явно моделирует наследование.

Таким образом Z может использоваться для удовлетворительного определения отдельных УО, но для того, чтобы можно было говорить о наследовании и совместимости, необходимы дополнительные возможности языка.

Наследование не поддерживается в Z. Оно может быть моделировано включением схем в схему состояния.

Определение наследования УО требует, чтобы подклассы были совместимы. Но не требует, чтобы подклассы были подтипами в Z. Обычно УО может сообщить свой фактический класс. Так как атрибут «фактический класс» всегда содержит фактический класс объекта, то подкласс не может сообщить класс своего суперкласса. Следовательно, подкласс не может демонстрировать при возврате значения атрибута «фактический класс» то же самое поведение, что и его суперкласс (т. е. они не взаимозаменяемы), даже если их поведение алломорфно. Следовательно, подклассы управляемых объектов не эквивалентны подтипам в Z, где подтип будет демонстрировать то же самое поведение, что и супертип. Однако подкласс демонстрирует очень мало «неподходящее» поведение.

Можно показать, что определенное в настоящем стандарте наследование УО позволяет специфицировать поведение родителя, которое будет несовместимо с поведением потомков. Так как такое неподходящее поведение очень ограничено, то класс УО может быть представлен двумя спецификациями классов. Одна из этих спецификаций охватывает поведение, которое может демонстрировать любой экземпляр любого расширения УО, другая — поведение, демонстрируемое только экземплярами совместимого класса, но не расширениями. Эта последняя спецификация является как раз той, которая реализуется для получения полного поведения фактического экземпляра УО.

В.3.6 Пакеты

Многие части функциональных возможностей класса могут присутствовать в одних конкретных УО и отсутствовать в других. В настоящем стандарте это описывается с помощью группирования функциональных возможностей в условные пакеты. Тогда каждый УО реализует подходящие пакеты. В Z функциональные возможности не могут быть предоставлены таким способом, но можно сделать поведение УО зависящим от того, какие пакеты реализованы. Это можно сделать непосредственным образом потому, что УО должен содержать атрибут административного управления, называемый «пакеты», в котором перечислены идентификаторы объектов фактически реализованных пакетов. Таким образом для моделирования поведения условного пакета само поведение становится зависящим от присутствия идентификатора пакета в атрибуте «пакеты».

В.3.7 Класс

Для определения класса УО необходимо представить его атрибуты и операции. Атрибуты становятся частью схемы состояния Z, а операции становятся схемами операций Z.

В.3.7.1 Атрибуты

Атрибуты управляемого объекта объявляются в схеме состояния. Каждый атрибут имеет заданный тип, который может быть типом, объявленным в части АСН. 1 шаблона РОУО или в полной формальной модели Z.

В.3.7.2 Операция Get

Управляющий может запросить выполнение операции Get над УО. В определении УО ИУМ-Get имеет много параметров, но большинство из них относится к управлению доступом, выбору объекта и т. п. В конкретном экземпляре Get может быть смоделирована на границе управляемого объекта, игнорируя указанные вопросы и заменяя единственную операцию Get несколькими операциями Get <имя, где <имя — единственный атрибут.

В.3.7.3 Операция GetAll

Операция GetAll, которая не имеет входных параметров, может быть смоделирована таким же образом. Он возвращает непустой набор значений атрибутов.

В.3.7.4 Операция Replace

Установка в конкретном УО запрашивается операцией УО ИУМ-Set. В настоящей спецификации операции Replace моделируются как видимые на границе УО. Эти операции относятся к установкам атрибутов, установкам значений по умолчанию, добавлению и удалению значений. Следовательно, для представления каждого из этих изменений должна быть специфицирована схема Z.

В.3.7.5 Сообщения

Являются незапрошенными посланиями УО для отчетов о событиях в УО. Они моделируются не как операции, а как результаты операций, происходящих над УО. Таким образом, любая операция (вызванная управляющим или внутренне, ресурсом) может создать выходной результат и, если операция приводит к сообщению, то оно будет частью выходного результата операции. Это означает, что выходным результатом схемы операции Z, которая может вызвать сообщения, должно быть множество сообщений. Случай, когда сообщение не создается, может быть представлен как пустое множество в качестве выходного результата.

Данные в сообщении состоят из типа события Event Type, который является идентификатором объекта его стандартного определения. Идентификатор объекта может быть определен как постоянная, конкретные данные — как тип схемы. Поведение сообщения включает в себя каждый объект, который может это сообщение создавать.

В.3.7.6 Действия

Являются операциями, осуществляемыми управляющим над УО. Они естественным образом представляются операциями Z.

В.3.8 Спецификация системы объектов

В оставшейся части настоящего приложения описано представление поведения единичного объекта. При рассмотрении создания и удаления объектов, связывания имен, вложения и наименования необходимо описать состояние системы, в которой находятся объекты. Создание и удаление объекта могут быть представлены как изменение состояния этой системы. Связывание имени вложение могут быть представлены отношением на множестве объектов. Наименование может быть определено терминах этого отношения.

В.4 Пример

В настоящем подразделе приведен пример определений высшего класса УО и атрибутов административного управления. Так как в данном руководстве основное внимание уделяется моделированию поведения, то этот подраздел не показан создание типов Z из типов АСН.1. Полное формальное определение дано в В.7.

В.4.1 Класс топ

Первым классом, который должен быть определен, является *top*, представляющий собой основного родителя (в иерархии наследования) для всех УО.

Класс *top* имеет четыре атрибута административного управления: *objectClass*, *packages*, *allomorphs* и *nameBinding*. В атрибуте *objectClass* хранится идентификатор объекта класса, в *packages* — идентификаторы объектов реализованных пакетов, в *nameBinding* — идентификатор объекта связывания имен, использованного при реализации объекта, а в *allomorphs* — идентификаторы объектов классов, с которыми объект может быть алломорфен. Так как атрибуты административного управления могут находиться в пакетах, атрибуты, присутствующие в УО данного класса, могут изменяться. Это моделируется включением дополнительного атрибута *attributes*, в котором хранятся идентификаторы объектов атрибутов, фактически реализованных в данном УО. Все атрибуты, присутствующие в классе *top*, фиксированы на протяжении жизни конкретного УО.

Интерфейсы в Z не моделируются явно и, таким образом, не возможно формально определить, какие операции вызываются внутренне управляющим, какие — внешне:

TopState

```
allomorphs : F O B J E C T I
objectClass : O B J E C T I
nameBinding : O B J E C T I
packages : F O B J E C T I
attributes : F O B J E C T I
```

```
{objectClassOid, nameBindingOid} ⊂ attributes
allomorphsPackageOid ∈ packages ⇒ allomorphsOid ∈ attributes
packagesPackageOid ∉ packages
packages ≠ ∅ ⇒ packagesOid ∈ attributes
```

attributes является не атрибутом УО, а новым компонентом состояния, определенным для удобства. В нем перечислены атрибуты, которые содержатся в УО. Следовательно, инвариант требует, чтобы он содержал идентификаторы объектов подходящих атрибутов, как описано в В.3.3 (и определено в В.7.4). Атрибуты *objectClass* и *nameBinding* являются обязательными. Атрибут *packages* присутствует только в том случае, когда реализован любой зарегистрированный пакет, отличный от *packagesPackage*. В последнем случае этот атрибут обозначает *allomorphsPackage*.

Операция *TopGetNameBinding* спрашивает УО и возвращает значение атрибута *nameBinding* без изменения *TopState*. Операция *TopGetNameBinding* вызывается управляющим:

TopGetNameBinding

```
∃ TopState
result† : O B J E C T I
result† : nameBinding
```

Операция *TopGetAllomorphs*, *TopGetObjectClass* и *TopGetPackages* здесь не определяются. Нет операции для получения *attributes*, так как *attributes* не является атрибутом УО, определенным в шаблоне РОУО.

Операция *TopGetAll* получит значения атрибутов объекта. Она всегда возвращает значения *objectClass* и *nameBinding*. Если присутствуют условные пакеты или алломорфизм, то возвращаются и они. Операция *TopGetAll* вызывается управляющим:

TopGetAll

$\exists TopState$ $result \vdash PAttributeValues$
$\# attributes = \# result \mid$ $ObjectClassValueobjectClass \in result \mid$ $NameBindingValuenameBinding \in result \mid$ $PackagesOid \in attributes \Rightarrow packagesValuepackages \in result \mid$ $AllomorphsOid \in attributes \Rightarrow allomorphsValueallomorphs \in result \mid$

Конструкция *TopEventReport* является способом моделирования сообщений. *TopEventReport* возникает спонтанно и представляет способ отчета в событиях, которые не контролируются управляющим:

TopEventReport

$\exists TopState$ $notification \vdash EventInfo$

В.4.2 Класс State Management

Не отражает никакой конкретный класс УО. Он отражает поведение любого объекта, содержащего любой из стандартных атрибутов *administrativeState*, *operationalState* и *usageState*. В рамках данного подхода он удобен для понимания включения этих атрибутов как наследования и не является примером для использования.

Схема состояния включает в себя определения *TopState* и предикаты и специфицирует некоторые дополнительные переменные и соединяющие предикаты:

StateManagementState

$TopState$ $administrativeState:$ $AdministrativeState$ $operationalState : OperationalState$ $usageState : UsageState$
$operationalState = disabled \Rightarrow usageState = idle$ $administrativeState = locked \Rightarrow usageState = idle$ $usageState = idle \Rightarrow administrativeState \neq shuttingown$

Класс *StateManagement* наследует операции от *Top*. Хотя нет способавстроить в *Z* наследование операций, непосредственным методом моделирования является повторное определение операций в терминах нового состояния. Предикаты состояния *StateManagement* следуют из определения функции *StateManagement* в ГОСТРИСО/МЭК 10165-2 и ГОСТРИСО/МЭК 10164-2.

Может быть легко определена операция *SMGetNameBinding*, так как она не оказывает действия на новые переменные состояния, объявленные в *StateManagementState*. Определение *TopGetNameBinding* может быть использовано повторно:

SMGetNameBinding

$TopGetNameBinding$ $\exists StateManagementState$

Операции для получения других атрибутов *StateManagementState* также опущены в этом примере. Для операций *SMGetAllomorphs*, *SMGetObjectClass* и *SMGetPackages* могут быть повторно использованы определения из *Top*, как для *SMGetNameBinding*. Необходимо определить новые операции *GetSMAdministrativeState*, *GetSMOperationalState* и *SMGetUsageState*. Можно повторно использовать *SMEventReport*.

Схема *SMGetAll* также использует операции, определенные в *TopState*. Она включает в себя определение *TopGetAll* и усиленные постулаты:

SMGetAll

$\exists StateManagementState$ $TopGetAll$
$administrativeStateOid \in attributes$ $\Rightarrow administrativeStateValueadministrativeState \in result \mid$ $OperationalStateOid \in attributes$ $\Rightarrow operationalStateValueoperationalState \in result \mid$ $UsageStateOid \in attributes$ $\Rightarrow usageStateValueusageState \in result \mid$

Операция *SMReplaceAdministrativeState* описывает поведение, специфичное для класса *StateManagement* путем замены административного состояния другим значениям, предоставленным в качестве входного. При

осуществлении операции в зависимости от состояния объекта может измениться и статус использования. Операционный статус этой операцией не изменяется:

SMReplaceAdministrativeState

Δ StateManagementState \exists TopState input?: AdministrativeState administrativeState' \in IF usageState \neq idle THEN { unlocked \rightarrow unlocked, locked \rightarrow locked, shuttingown \rightarrow locked, locked \rightarrow shuttingown, shuttingown \rightarrow shuttingown } ({input?}) ELSE { unlocked \rightarrow unlocked, locked \rightarrow locked, shuttingown \rightarrow locked } ({input?}) administrativeState' \in locked \rightarrow usageState' = idle administrativeState' \neq locked \rightarrow usageState' = usageState operationalState' = operationalState

Поведение, специфицированное в предикатах схемы, является формализацией неформального описания в ГОСТРИСО/МЭК 10164-2. Для полноты должны быть определены операции замены операционного статуса и статуса использования.

Существует ряд других операций, которые описывают поведение, специфичное для класса StateManagement. Эти операции не перечислены здесь, но приведены в В.7. В их число входят *SMCapacityDecrease*, *SMCapacityIncrease*, *SMisable*, *SMinable*, *SMNewUser* и *SMUserQuit*.

В.4.3 Реализуемые классы

Ни один из описанных выше классов не может быть реализован. Используемая процедура построения классов может быть продолжена. Класс *StateManagement* можно повторно использовать для определения класса, названного *CIRCUIT*, который, в свою очередь, может быть использован для определения класса *ECIRCUIT* и, следовательно, реализуемого класса *ActualECircuit*. Эта часть работы не приведена в руководстве, так как процедура точно такая же, что и описанная выше, и повторение не добавит ничего нового.

В.5 Нерассмотренные вопросы

В данном подразделе перечислены основные вопросы, встречающиеся при переводе спецификаций управляемых объектов РОУОВ Z. Также приведена предлагаемая неформальная трактовка тех относящихся к Z случаев, когда нет соответствующих конструкций для конкретных характеристик спецификации управляемого объекта.

В.5.1 Определение поведения в управляемых объектах

В шаблонах термин «определение поведения» используется почти для всех категорий, являющихся данными или обработкой. В последнем случае это определение может включать всебя информацию о фактическом поведении (в точном смысле этого слова) или статическую информацию о категории, такую как ее назначение, или что-то другое. При переводе необходимо проанализировать текст, приведенный под указанным заголовком, и извлечь из него информацию, относящуюся к поведению рассматриваемой категории. Эта информация будет использоваться при формальном переводе, а имеющийся текст может быть включен в спецификацию Z в качестве комментария.

В.5.2 Внутренние операции в Z

Внутренняя операция управляемого объекта представляет случай, когда сообщение создается спонтанно (без участия вызова административного управления). Внутренние операции являются желательной характеристикой многих систем. В настоящее время в Z эта характеристика представляется неформально с помощью комментария на естественном языке.

В.5.3 Абстрактные классы в Z

Иногда полезно идентифицировать абстрактные классы, т.е. классы, которые не имеют своей собственной реализации. Некоторые классы УО (например, top) не могут быть реализованы. Было бы полезно показать, как и часть спецификации Z представляют классы, которые не могут быть реализованы. В настоящее время это делается с помощью неформальной аннотации.

В.5.4 Семантика конструкции PARAMETERS

Включение семантики конструкции PARAMETERS в объекты не рассматривается в настоящем стандарте.

В.6 Преобразование типов данных АСН.1 в Z

Ниже рассмотрены вопросы перевода конструкций АСН.1.

В.6.1 Простые типы

АСН.1 включает в себя некоторые простые типы, которые являются встроенными. Они имеют стандартную структуру, но обычно это не представляет интереса в контексте настоящего стандарта и их следует представлять как заданное множество. Имеется большой набор типов символьных строк:

[*NUMERICSTRING, PRINTABLESTRING, TELETEXSTRING, VIEOTEXSTRING, VISIBLESTRING, IA5STRING, GRAPHICSTRING, GENERALSTRING*]

Два из них имеют синонимы:

T61STRING = = *TELETEXSTRING*
ISO64STRING = = *VISIBLESTRING*

Из других простых типов целый может быть представлен как *Z*, а булевский и вырожденный — как свободные типы:

Boolean : : = *btrue* | *bfalse*
Null : : = *null*

Эти три типа определяют нотацию их значений.

Вещественный тип, строки битов и октетов могут быть представлены как заданные множества (хотя иногда для строк битов и октетов может потребоваться структура):

[*REAL, BITSTRING, OCTETSTRING*]

В настоящем стандарте описан еще один специальный тип, который может быть представлен как заданное множество:

[*OBJECT*]

Он представляет идентификатор объекта АСН.1.

Идентификаторы объектов фактически являются непустыми последовательностями из *N* и, вместо заданных множеств, могут быть смоделированы именами. В некоторых случаях соответствующей нотации значения должен быть придан некоторый смысл.

В АСН.1 определено несколько «полезных» типов. Хотя они могут быть определены в терминах других конструкций АСН.1, удобно представлять их в виде заданных множеств:

[*GENERALIZETIME, UTCTIME, OBJECTESCRIPTOR, EXTERNAL*]

ANY

В АСН.1 имеется специальный тип *ANY*, который может содержать АСН.1 любого другого типа. Такой тип недопустим в *Z* и было бы трудно расширить этот язык для включения подобного типа. Однако, задавая некоторое известное множество типов, можно определить свободный тип *Z*, который может включать в себя любой из этих типов. Альтернативный подход состоит в том, чтобы определить *ANY* как заданное множество для целей проверки типа. Это приемлемо до тех пор, пока нам не нужно делать ничего другого. Тип *AttributeValues* обычно замещает *ANY*, что описано ниже.

В.6.2 Структурированные типы

Другие типы АСН.1 строятся на основе конструкций.

Множество

Множества АСН.1 могут быть представлены в *Z* либо как кортежи, либо как схемы. Кортэжи *Z* не позволяют именовать компоненты, и в этом отношении больше подходят схемы. Однако нотация значений *Z* для схем менее удобна. Тегирование в *Z* невозможно и не нужно, так как компоненты множеств всегда могут быть опознаны либо по их положению в кортеже, либо по имени компонента в схеме.

Компоненты этого и других структурированных типов могут быть факультативными (*OPTIONAL*). Это может быть представлено в *Z* дополнением типа факультативного компонента специальным значением «отсутствует». Значения по умолчанию *EFAULT* не могут быть представлены удобным образом как свойства типа данных. Можно представить поведение, подразумеваемое умолчанием, во всех операциях над этим типом данных.

Последовательность

Последовательности АСН.1 могут моделироваться точно таким же образом, как множества АСН.1, так как единственным различием является явно установленный порядок. Поэтому более подходящими являются кортежи, но можно использовать и схемы.

Множество-из

Множества-из АСН.1 могут моделироваться как последовательности *Z*.

Выбор

Выборочные типы АСН.1 непосредственно являются перечислениями и могут моделироваться свободными типами *Z*.

С этим типом связана серьезная проблема области действия. В АСН.1 конструкции в пределах выборочного типа являются локальными для этого типа. Таким образом одно и то же имя конструкции может использоваться в нескольких перечислениях. В *Z* имена являются глобальными и могут быть повторно использованы. Эта проблема обычно может быть разрешена изменением имен конструкций, как правило, путем добавления к ним в качестве префикса имени их типа.

Аналогичная проблема возникает, когда создаются типы АСН.1, являющиеся синонимами, например целого, но с поименованными значениями. Эти поименованные значения являются локальными для типа-синонима в АСН.1, но глобальными синонимами для целых в *Z*. Для разрешения этой проблемы также можно изменить имена конструкций.

В.6.3 Простые типы атрибутов

Проще всего представить атрибут в УО как переменную Z соответствующим типом данных. Потребуется также определение постоянной, представляющей *OBJECTI* этого атрибута. Когда для атрибута определены операции согласования, может быть использовано фактическое стандартное значение параметра для согласования.

Рассмотрим атрибут *AdministrativeState*. Имеется определение типа:

AdministrativeState : := *unlocked* | *locked* | *shutting down*

Может быть определена постоянная для представления идентификатора объекта атрибута:

administrativeStateOid : *OBJECTI*

Фактическое значение идентификатора может быть представлено как ограничение на это аксиоматическое определение.

Может быть определена переменная состояния в УО:

MOState

administrativeState: *AdministrativeState*

Тако решение является прямым и удобным, но требует списка аксиоматических определений *OBJECTI*. Оно также делает связь между именем атрибута и его *OBJECTI* чисто синтаксической. В этом решении было принято соглашение об использовании суффикса *Oid*.

В.6.4 Типы атрибутов как схемы

Можно включить все эти характеристики атрибута в один тип схемы, который будет типом переменной Z , моделирующей атрибут УО:

AdministrativeStateType

value: *AdministrativeState*

Oid: *OBJECTI*

Oid = <4, 3, 19, 27, 1, 3

Важно подставить именно здесь значение *OBJECTI*, так как необходимо гарантировать, что оно не может быть изменено.

Структура *OBJECTI* пока не определена, но запись

OBJECTI = = *seqIN*

является одной из возможностей придать смысл предыдущей схеме. Эта же схема может хранить параметр для согласования, если окажется важным представить его в спецификации.

Теперь УО может содержать атрибут этого типа:

MOState

administrativeState: *AdministrativeStateType*

Ссылка на его значение (или на его *Oid*) может быть сделана через выбор компонента, например *administrativeState.value*.

Этот метод удобен, но передает семантику для связи между атрибутом и его *OBJECTI*. Однако для читателей спецификации может показаться странным, что *Oid* присутствует в состоянии УО, хотя он и не может изменяться (и фактически является глобальной постоянной, известной на момент спецификации).

В.6.5 Тип Attribute Values

Как уже отмечалось, в Z трудно моделировать тип АСН.1 ANY. Общим подходом является задание списков значений атрибута. Требуется определение свободного типа Z комбинации суженных типов. Этот подход работает до тех пор, пока множество используемых атрибутов фиксировано на момент спецификации. Тогда решение будет выглядеть приблизительно так:

AttributeValues : := *administrativeStateValue* << *AdministrativeState* |

objectClassValue << *OBJECTI* |

nameBindingValue << *OBJECTI* |

packagesValue << *POBJECTI* |

allomorphsValue << *POBJECTI* |

operationalStateValue << *OperationalState* |

usageStateValue << *UsageState*

В.7 Полный пример

В настоящем подразделе представлена полная формальная модель, на которой основан пример в В.4. Модель представлена в традиционном для Z стиле объявления до использования; а именно, определения типов, преобразованные из АСН.1, появляются в начале, а определения поведения — в конце.

Следует прокомментировать одно место в стиле спецификации. Определения *AttributeValues* и *OBJECTINSTANCES* являются взаиморекурсивными. В Z это технически недопустимо, и для того, чтобы дать определения, было сделано следующее. Тип *AttributeValues* был введен как заданное множество. Затем

ГОСТРИСО/МЭК10165-4—2001

OBJECTINSTANCE было определено с использованием заданного множества *AttributeValues*. Определение *OBJECTINSTANCE* было использовано для введения ограничений того, что может принимать *AttributeValues*. Была выполнена проверка ограничения для того, чтобы показать, что такие множества фактически существуют, но в примере это не показано.

В.7.1 Основные типы АСН.1

[*NUMERICSTRING*, *PRINTABLESTRING*, *TELETEXSTRING*, *VIEOTEXSTRING*]

[*VISIBLESTRING*, *IA5STRING*, *GRAPHICSTRING*, *GENERALSTRING*]

T61STRING = = *TELETEXSTRING*

ISO64STRING = = *VISIBLESTRING*

Boolean : : = *btrue* | *bfalse*

Null : : = *null*

[*REAL*, *BITSTRING*, *OCTETSTRING*]

[*OBJECTI*]

[*ANY*]

[*GENERALIZETIME*, *UTCTIME*, *OBJECTESRIPTOR*, *EXTERNAL*]

В.7.2 Атрибуты УО

Следующее заданное множество резервирует место для более сложного и полного определения свободного типа, которое будет расширяться по мере определения новых классов.

[*AttributeValues*]

AttributeValuesOptional : : = *present* << *AttributeValues* | *absent*

RelativeistinguishedName = = *AttributeValues*

RNSequence = = *seq* *RelativeistinguishedName*

istinguishedName = = *RNSequence*

OBJECTINSTANCE : : = *distinguishedName* << *istinguishedName* | *nonSpecificForm* << *N*
| *localistinguishedName* << *RNSequence*

В.7.3 Сообщения

ProbableCause = = *OBJECTI*

SpecificIdentifier : : = *globalvalue* << *OBJECTI* | *localValue* << *N*

SpecificProblems = = *P* *SpecificIdentifier*

SpecificProblemsOptional : : = *sPPresent* << *SpecificProblems* | *sPAbsent*

PerceivedSeverity : : = *indeterminate* | *critical* | *major* | *minor* | *warning* | *cleared*

BackedUpStatus = = *Boolean*

BackedUpStatusOptional : : = *bUSPresent* << *BackedUpStatus* | *bUSAbsent*

ObjectInstanceOptional : : = *oISPPresent* << *OBJECTINSTANCE* | *oIAbsent*

TrendIndication : : = *lessSevere* | *noChange* | *moreSevere*

TrendIndicationOptional : : = *tIPresent* << *TrendIndication* | *tIAbsent*

ObservedValue : : = *int* << *N* | *real* << *REAL*

ObservedValueOptional : : = *oVPresent* << *ObservedValue* | *oVAbsent*

ThresholdLevelInd : : =

up << *ObservedValue* · *ObservedValueOptional*

| *down* << *ObservedValue* · *ObservedValueOptional*

ThresholdLevelIndOptional : : = *tLIPresent* << *ThresholdLevelInd* | *tLIAbsent*

ArmTimeOptional : : = *aTPresent* << *GENERALIZETIME* | *aTAbsent*

ThresholdInfo = =

OBJECTI · *ObservedValue* · *ThresholdLevelIndOptional* · *ArmTimeOptional*

ThresholdInfoOptional : : = *thIPresent* << *ThresholdInfo* | *thIAbsent*

NotificationIdentifier = = *N*

NotificationIdentifierOptional : : = *nIPresent* << *NotificationIdentifier* | *nIAbsent*

CorrelatedNotifications = = *P* ((*P* *NotificationIdentifier*) · *ObjectInstanceOptional*)

CorrelatedNotificationsOptional : : = *cNPresent* << *CorrelatedNotifications* | *cNAbsent*

AttributeValueChangeefinition = = *P* (*OBJECTI* · *AttributeValuesOptional* · *AttributeValues*)

AttributeValueChangeefinitionOptional : : =

aVCPresent << *AttributeValueChangeefinition* | *aVCAbsent*

MonitoredAttributes = = *P* *OBJECTI*

MonitoredAttributesOptional : : = *mAPresent* << *MonitoredAttributes* | *mAAbset*

ProposedRepairActions = = *P* *SpecificIdentifier*

ProposedRepairActionsOptional : : = *pRAPresent* << *ProposedRepairActions* | *pRAAbset*

AdditionalTextOptional : : = *adTPresent* << *GRAPHICSTRING* | *adTAbsent*

ManagementExtension = = *OBJECTI* · *Boolean* · *ANY*

AdditionalInformation = = *P* *ManagementExtension*

AdditionalInformationOptional : : = *aIPresent* << *AdditionalInformation* | *aIAbsent*

SourceIndicator : : = *resourceOperation* | *managementOperation* | *sIUnknown*

SourceIndicatorOptional: : = *sIPresent* << *SourceIndicator* | *sIAbsent*
AttributeIdentifierList = = **P** *OBJECTI*
AttributeIdentifierListOptional: : = *atIPresent* << *AttributeIdentifierList* | *atIAbsent*
Attribute = = **O** *OBJECTI* · *AttributeValues*
AttributeList = = **P** *Attribute*
AttributeListOptional: : = *aLPresent* << *AttributeList* | *aLAbsent*

AlarmInfo

probableCause: *ProbableCause*
specificProblems: *SpecificProblemsOptional*
perceivedSeverity: *PerceivedSeverity*
backedUpStatus: *BackedUpStatusOptional*
backUpObject: *ObjectInstanceOptional*
trendIndication: *TrendIndicationOptional*
thresholdInfo: *ThresholdInfoOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
stateChangeefinition: *AttributeValueChangeefinitionOptional*
monitoredAttributes: *MonitoredAttributesOptional*
proposedRepairActions: *ProposedRepairActionsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

AttributeValueChangeInfo

sourceIndicator: *SourceIndicatorOptional*
attributeIdentifierList: *AttributeIdentifierListOptional*
attributeValueChangeefinition: *AttributeValueChangeefinitionOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

ObjectInfo

sourceIndicator: *SourceIndicatorOptional*
attributeList: *AttributeListOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

RelationshipChangeInfo

sourceIndicator: *SourceIndicatorOptional*
attributeIdentifierList: *AttributeIdentifierListOptional*
relationshipChangeefinition: *AttributeValueChangeefinitionOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

SecurityAlarmInfo

notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

StateChangeInfo

sourceIndicator: *SourceIndicatorOptional*
attributeIdentifierList: *AttributeIdentifierListOptional*
stateChangeefinition: *AttributeValueChangeefinitionOptional*
notificationIdentifier: *NotificationIdentifierOptional*
correlatedNotifications: *CorrelatedNotificationsOptional*
additionalText: *AdditionalTextOptional*
additionalInformation: *AdditionalInformationOptional*

EventInfo: := *attributeValueChange*<<*AttributeValueChangeInfo*

| *communicationsAlarm*<<*AlarmInfo*
 | *environmentalAlarm*<<*AlarmInfo*
 | *equipmentAlarm*<<*AlarmInfo*
 | *integrityViolation*<<*SecurityAlarmInfo*
 | *objectCreation*<<*ObjectInfo*
 | *objectDeletion*<<*ObjectInfo*
 | *operationalViolation*<<*SecurityAlarmInfo*
 | *physicalViolation*<<*SecurityAlarmInfo*
 | *processingError*<<*AlarmInfo*
 | *qualityOfServiceAlarm*<<*AlarmInfo*
 | *relationshipChange*<<*RelationshipChangeInfo*
 | *securityServiceOrMechanismViolation*<<*SecurityAlarmInfo*
 | *stateChange*<<*StateChangeInfo*
 | *timeomainViolation*<<*SecurityAlarmInfo*

B.7.4 «CCITTRec. X.721(1992)»|ISO/IEC 10165-2:1992:Top

allomorphsOid : OBJECTI
nameBindingOid : OBJECTI
objectClassOid : OBJECTI
packagesOid : OBJECTI

allomorphsValue : (FOBJECTI) → AttributeValues
nameBindingValue : OBJECTI → AttributeValues
objectClassValue : OBJECTI → AttributeValues
packagesValue : (FOBJECTI) → AttributeValues

disjoint<*ranallomorphsValue*,*rannameBindingValue*,
ranobjectClassValue,*ranpackagesValue*

packagesPackageOid: OBJECTI
allomorphsPackageOid : OBJECTI

TorState

allomorphs : F OBJECTI
objectClass : OBJECTI
nameBinding : OBJECTI
packages : F OBJECTI
attributes : F OBJECTI

{*objectClassOid*,*nameBindingOid*} ⊂ *attributes*
allomorphsPackageOid ∈ *packages* ⇒ *allomorphsOid* ∈ *attributes*
packagesPackageOid ∉ *packages*
packages ≠ ∅ ⇒ *packagesOid* ∈ *attributes*

TopGetNameBinding

∃ *TopState*
result : OBJECTI
result = *nameBinding*

TopGetAll

∃ *TopState*
result : PAttributeValues

attributes = # *result* |
ObjectClassValueobjectClass ∈ *result* |
NameBindingValuenameBinding ∈ *result* |
PackagesOid ∈ *attributes* ⇒ *packagesValuepackages* ∈ *result* |
AllomorphsOid ∈ *attributes* ⇒ *allomorphsValueallomorphs* ∈ *result* |

TopEventReport

 \exists *TopState*
notification | : *EventInfo*

B.7.5 Класс State Management

administrativeStateOid: OBJECTI
operationalStateOid: OBJECTI
usageStateOid: OBJECTI

AdministrativeState: := *unlocked* | *locked* | *shuttingown**OperationalState*: := *enabled* | *disabled**UsageState*: := *idle* | *active* | *busy*

administrativeStateValue: *AdministrativeState* → *AttributeValues*
operationalStateValue: *OperationalState* → *AttributeValues*
usageStateValue: *UsageState* → *AttributeValues*

disjoint < *ranallomorphsValue*, *rannameBindingValue*,
ra nobjectClassValue, *ranpackagesValue*,
ra nadministrativeStateValue,
ra noperationalStateValue, *ranusageStateValue*

StateManagementState

TopState
administrativeState:
AdministrativeState
operationalState: *OperationalState*
usageState: *UsageState*

operationalState = *disabled* ⇒ *usageState* = *idle*
administrativeState = *locked* ⇒ *usageState* = *idle*
usageState = *idle* ⇒ *administrativeState* ≠ *shuttingown*

SMGetNameBinding

TopGetNameBinding
 \exists *StateManagementState*

SMEventReport

TopEventReport
 \exists *StateManagementState*

SMGetAll

 \exists *StateManagementState*
TopGetAll

administrativeStateOid ∈ *attributes*
⇒ *administrativeStateValue* *administrativeState* ∈ *result* |
OperationalStateOid ∈ *attributes*
⇒ *operationalStateValue* *operationalState* ∈ *result* |
UsageStateOid ∈ *attributes*
⇒ *usageStateValue* *usageState* ∈ *result* |

SMCapacityincrease

 Δ *StateManagementState*
 \exists *TopState*

usageState = *active* ⇒ *usageState'* ∈ {*active*, *busy*}
usageState ≠ *active* ⇒ *usageState'* = *usageState*
administrativeState' = *administrativeState*
operationalState' = *operationalState*

SMCapacityIncrease

Δ StateManagementState
 \exists TopState

$usageState = busy \Rightarrow usageState' = active$
 $usageState \neq busy \Rightarrow usageState' = usageState$
 $administrativeState' = administrativeState$
 $operationalState' = operationalState$

SMisable

Δ StateManagementState
 \exists TopState

$administrativeState' =$
IF $administrativeState = shuttingown$
THEN $locked$
ELSE $administrativeState$
 $operationalState' = disabled$
 $usageState' = idle$

SMEnable

Δ StateManagementState
 \exists TopState

$operationalState = disabled$
 $operationalState' = enabled$
 $administrativeState' = administrativeState$
 $usageState' = usageState$

SMNewUser

Δ StateManagementState
 \exists TopState

$operationalState = enabled$
 $administrativeState = unlocked$
 $usageState \in \{idle, active\}$
 $usageState' \in \{active, busy\}$
 $administrativeState' = administrativeState$
 $operationalState' = operationalState$

SMUserQuit

Δ StateManagementState
 \exists TopState

$administrativeState = shuttingown \Rightarrow usageState' = idle$
 $\Rightarrow administrativeState' = locked$
 $administrativeState \neq shuttingown \Rightarrow usageState' \neq idle$
 $\Rightarrow administrativeState' = administrativeState$
 $usageState \in \{active, busy\}$
 $usageState' \in \{idle, active\}$
 $operationalState' = operationalState$

SMReplaceAdministrativeState

Δ StateManagementState
 \exists TopState

$administrativeState' \in$
IF $usageState \neq idle$
THEN { $unlocked \rightarrow unlocked, locked \rightarrow locked,$
 $shuttingown \rightarrow locked, locked \rightarrow shuttingown,$
 $shuttingown \rightarrow shuttingown$ } ($\{input?\}$)
ELSE { $unlocked \rightarrow unlocked, locked \rightarrow locked,$
 $shuttingown \rightarrow locked$ } ($\{input?\}$)
 $administrativeState' = locked \rightarrow usageState' = idle$
 $administrativeState' \neq locked \rightarrow usageState' = usageState$
 $operationalState' = operationalState$

УДК 681.324:006.354

ОКС 35.100.70

П 85

ОКСТУ 4002

Ключевые слова: информационная технология, взаимосвязь открытых систем, обработка данных, информационный обмен, сетевое взаимодействие, административное управление, информация

Редактор *В. П. Огурцов*
Технический редактор *Н. С. Гришанова*
Корректор *Н. И. Гаврищук*
Компьютерная верстка *Т. Ф. Кузнецовой*

Изд. лиц. № 02354 от 14.07.2000. Сдан в набор 25.09.2001. Подписан в печать 16.11.2001. Усл. печ. л. 7,90. Уч. - изд. л. 8,00.
Тираж 429 экз. С 2811. Зак. 2300

ИПК Издательство стандартов, 107076, Москва, Колодезный пер., 14.
<http://www.standards.ru> e-mail: info@standards.ru
Набрано в Калужской типографии стандартов на ПЭВМ.
Калужская типография стандартов, 248021, Калуга, ул. Московская, 256.
ПЛР № 040138