МЕЖДУНАРОДНЫЙ СОЮЗ ЭЛЕКТРОСВЯЗИ

МСЭ-Т X.511

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

(08/2005)

СЕРИЯ X: СЕТИ ПЕРЕДАЧИ ДАННЫХ И
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
И БЕЗОПАСНОСТЬ

Справочник

Информационная технология – Взаимосвязь открытых систем – Справочник: Определение абстрактной службы

Рекомендация МСЭ-Т X.511

РЕКОМЕНДАЦИИ МСЭ-Т СЕРИИ X

СЕТИ ПЕРЕДАЧИ ДАННЫХ, ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ И БЕЗОПАСНОСТЬ

СЕТИ ПЕРЕДАЧИ ДАННЫХ ОБЩЕГО ПОЛЬЗОВАНИЯ	
Службы и услуги	X.1–X.19
Интерфейсы	X.20–X.49
Передача, сигнализация и коммутация	X.50–X.89
Сетевые аспекты	X.90–X.149
Техническое обслуживание	X.150–X.179
Административные предписания	X.180–X.199
ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ	
Модель и обозначение	X.200–X.209
Определения служб	X.210–X.219
Спецификации протоколов с установлением соединений	X.220–X.229
Спецификации протоколов без установления соединений	X.230–X.239
Проформы PICS	X.240–X.259
Идентификация протоколов	X.260–X.269
Протоколы обеспечения безопасности	X.270–X.279
Управляемые объекты уровня	X.280–X.289
Испытание на соответствие	X.290–X.299
ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕТЯМИ	
Общие положения	X.300–X.349
Спутниковые системы передачи данных	X.350–X.369
Сети, основанные на протоколе Интернет	X.370–X.379
СИСТЕМЫ ОБРАБОТКИ СООБЩЕНИЙ	X.400–X.499
СПРАВОЧНИК	**X.500–X.599**
ОРГАНИЗАЦИЯ СЕТИ ВОС И СИСТЕМНЫЕ АСПЕКТЫ	
Организация сети	X.600–X.629
Эффективность	X.630–X.639
Качество обслуживания	X.640–X.649
Наименование, адресация и регистрация	X.650–X.679
Абстрактно-синтаксическая нотация 1 (ASN.1)	X.680–X.699
УПРАВЛЕНИЕ В ВОС	
Структура и архитектура управления системами	X.700–X.709
Служба и протокол связи для общего управления	X.710–X.719
Структура управляющей информации	X.720–X.729
Функции общего управления и функции ODMA	X.730–X.799
БЕЗОПАСНОСТЬ	X.800–X.849
ПРИЛОЖЕНИЯ ВОС	
Фиксация, параллельность и восстановление	X.850–X.859
Обработка транзакций	X.860–X.879
Удаленные операции	X.880–X.889
Общие приложения ASN.1	X.890–X.899
ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА	X.900–X.999
БЕЗОПАСНОСТЬ ЭЛЕКТРОСВЯЗИ	X.1000–

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

МЕЖДУНАРОДНЫЙ СТАНДАРТ ИСО/МЭК 9594-3
РЕКОМЕНДАЦИЯ МСЭ-Т X.511

Информационная технология – Взаимосвязь открытых систем – Справочник: Определение абстрактной службы

Резюме

Настоящая Рекомендация | Международный стандарт определяет абстрактным способом обеспечиваемую Справочником службу, видимую внешнему пользователю, включая операции привязывания и отвязывания, чтения, поиска, модификации и обработки ошибок.

Источник

Рекомендация МСЭ-Т X.511 была утверждена 29 августа 2005 года 17-й Исследовательской комиссией МСЭ-Т (2005–2008 гг.) в соответствии с процедурой, изложенной в Рекомендации МСЭ-Т A.8. Идентичный текст опубликован также как Стандарт ИСО/МЭК 9594-3.

ПРЕДИСЛОВИЕ

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

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

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

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

ПРИМЕЧАНИЕ

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

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

ПРАВА ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

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

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

СОДЕРЖАНИЕ

Введение

Эта Рекомендация | Международный стандарт вместе с другими Рекомендациями | Международными стандартами была разработана для облегчения взаимосвязи систем обработки информации, которые должны обеспечивать справочные службы. Совокупность таких систем, вместе с хранимой ими справочной информацией, можно рассматривать как объединенное целое, называемое *Справочником (Справочной системой)*. Информация, хранимая в Справочнике, называемая в совокупности Информационной базой Справочника, ИБС (DIB, Directory Information Base), обычно используется для облегчения связи между объектами, с объектами или относительно объектов; примерами объектов могут служить прикладные процессы, люди, терминалы и списки рассылки.

Справочник играет существенную роль во Взаимосвязи открытых систем, его назначение заключается (при минимальных технических соглашениях вне самих стандартов взаимосвязи) в обеспечении взаимосвязи систем обработки информации:

- поставляемых разными производителями;
- находящихся под различным административным управлением;
- различных уровней сложности;
- различных поколений.

В настоящей Рекомендации | Международном стандарте определяются возможности, которые Справочник обеспечивает для своих пользователей.

В настоящей Рекомендации | Международном стандарте содержатся фундаментальные основы, на базе которых другие группы по разработке стандартов и отраслевые форумы могут определить отраслевые профили. Многие из свойств, определенные в этих основах как факультативные, могут быть сделаны обязательными для использования в некоторых средах посредством профилей. Это пятое издание технически пересматривает и расширяет, но не заменяет, четвертое издание этой Рекомендации | Международного стандарта. Реализации могут все еще соответствовать четвертому изданию. Однако в некоторый момент четвертое издание не будет поддерживаться (т. е. сообщаемые дефекты не будут исправляться). Рекомендуется, чтобы реализации соответствовали настоящему пятому изданию как можно скорее.

Настоящее пятое издание определяет версии 1 и 2 протоколов Справочника.

Первое и второе издания определяли только версию 1. Большинство услуг и протоколов, определенных в этом издании, разработано для работы с версией 1. Однако некоторые усовершенствованные службы и протоколы, например, подписанные параметры ошибок, не будут функционировать, если не все объекты Справочника, вовлеченные в операцию, имеют согласованную версию 2. Какая бы версия ни была согласована, различия между службами и различия между протоколами, определенными в пяти изданиях, кроме тех, которые специально предназначены для версии 2, примиряются с помощью правил расширяемости, которые определены в Рекомендации МСЭ-Т X.519 | ИСО/МЭК 9594-5.

В Приложении A, которое является составной частью настоящей Рекомендации | Международного стандарта, приводится модуль на языке ASN.1 для абстрактной службы Справочника.

В Приложении B, которое не является составной частью этой Рекомендации | Международного стандарта, приводятся диаграммы, которые описывают семантику, связанную с Управлением базовым доступом, в том виде, в котором она применяется для обработки операций Справочника.

В Приложении C, которое не является составной частью этой Рекомендации | Международного стандарта, даются примеры использования семейств статей.

В Приложении D, которое не является составной частью этой Рекомендации | Международного стандарта, перечисляются поправки и сообщения о дефектах, которые включены в состав этого издания Рекомендации | Международного стандарта.

МЕЖДУНАРОДНЫЙ СТАНДАРТ
РЕКОМЕНДАЦИЯ МСЭ-Т

Информационная технология – Взаимосвязь открытых систем – Справочник: Определение абстрактной службы

1 Сфера применения

Настоящая Рекомендация | Международный стандарт определяет абстрактным способом обеспечиваемую Справочником службу, видимую внешнему пользователю.

Настоящая Рекомендация | Международный стандарт не определяет конкретные реализации или разработки.

2 Нормативные справочные документы

Нижеследующие Рекомендации и Международные стандарты содержат положения, которые путем ссылки на них в данном тексте образуют положения настоящей Рекомендации | Международного стандарта. На момент публикации указанные издания были действительны. Все Рекомендации и Стандарты подвергаются пересмотру, поэтому сторонам соглашений, основанных на данной Рекомендации | Международном стандарте, следует рассматривать возможность применения самых последних изданий перечисленных ниже Рекомендаций и Стандартов. Члены МЭК и ИСО ведут регистры действующих в настоящее время Международных стандартов. Бюро стандартизации электросвязи МСЭ ведет список действующих в настоящее время Рекомендаций МСЭ-Т.

2.1 Идентичные Рекомендации | Международные стандарты

- Рекомендация МСЭ-Т X.200 (1994 г.) | ИСО/МЭК 7498-1:1994, *Информационные технологии – Взаимосвязь открытых систем – Базовая эталонная модель: Базовая модель.*
- Рекомендация МСЭ-Т X.500 (2005 г.) | ИСО/МЭК 9594-1:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Обзор концепций, моделей и служб.*
- Рекомендация МСЭ-Т X.501 (2005 г.) | ИСО/МЭК 9594-2:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Модели.*
- Рекомендация МСЭ-Т X.509 (2005 г.) | ИСО/МЭК 9594-8:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Структуры сертификатов открытого ключа и атрибута.*
- Рекомендация МСЭ-Т X.518 (2005 г.) | ИСО/МЭК 9594-4:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Процедуры распределенных операций.*
- Рекомендация МСЭ-Т X.519 (2005 г.) | ИСО/МЭК 9594-5:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Спецификации протоколов.*
- Рекомендация МСЭ-Т X.520 (2005 г.) | ИСО/МЭК 9594-6:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Избранные типы атрибутов.*
- Рекомендация МСЭ-Т X.521 (2005 г.) | ИСО/МЭК 9594-7:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Избранные классы объектов.*
- Рекомендация МСЭ-Т X.525 (2005 г.) | ИСО/МЭК 9594-9:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Репликация.*
- Рекомендация МСЭ-Т X.530 (2005 г.) | ИСО/МЭК 9594-10:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Использование административного управления системами для администрирования Справочника.*
- Рекомендация МСЭ-Т X.680 (2002 г.) | ИСО/МЭК 8824-1:2002, *Информационные технологии – Абстрактно-синтаксическая нотация версии 1 (ASN.1): Спецификация базовой нотации.*
- Рекомендация МСЭ-Т X.681 (2002 г.) | ИСО/МЭК 8824-2:2002, *Информационные технологии – Абстрактно-синтаксическая нотация версии 1 (ASN.1): Спецификация информационных объектов.*
- Рекомендация МСЭ-Т X.682 (2002 г.) | ИСО/МЭК 8824-3:2002, *Информационные технологии – Абстрактно-синтаксическая нотация версии 1 (ASN.1)*: *Спецификация ограничений*.

– Рекомендация МСЭ-Т X.683 (2002 г.) | ИСО/МЭК 8824-4:2002, *Информационные технологии – Абстрактно-синтаксическая нотация версии 1 (ASN.1)*: *Параметризация спецификаций ASN.1*.

2.2 Другие справочные материалы

– RFC 2025 (1996), *The Simple Public-Key GSS-API Mechanism (SPKM)*.

– RFC 2222 (1997), *Simple Authentication and Security Layer (SASL)*.

3 Определения

В настоящей Рекомендации | Международном стандарте применяются следующие определения терминов.

3.1 Определения основных терминов по Справочнику

Следующие термины определены в Рекомендации МСЭ-Т X.500 | ИСО/МЭК 9594-1:

a) *Справочник, Справочная система* (*Directory*);

b) *Информационная база Справочника* (*Directory Information Base*);

c) *пользователь (Справочника)* (*(Directory) User*).

3.2 Определения терминов по модели Справочника

Следующие термины определены в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2:

a) *системный агент Справочника* (*Directory System Agent*);

b) *агент пользователя Справочника* (*Directory User Agent*).

3.3 Определения терминов по Информационной базе Справочника

Следующие термины определены в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2:

a) *статья псевдонима* (*alias entry*);

b) *Информационное дерево Справочника* (*Directory Information Tree*);

c) *статья (Справочника)* (*(Directory) entry*);

d) *непосредственно предшествующая* (*immediate superior*);

e) *непосредственно предшествующая(ий) статья/объект* (*immediately superior entry/object*);

f) *объект* (*object*);

g) *класс объектов* (*object class*);

h) *статья объекта* (*object entry*);

i) *подчиненный младший* (*subordinate*);

j) *предшествующий, старший* (*superior*);

k) *порождающий элемент* (*ancestor*);

l) *семейство (статей)* (*family (of entries)*);

m) *составная статья* (*compound entry*).

3.4 Определения терминов по статьям Справочника

Следующие термины определены в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2:

a) *атрибут* (*attribute*);

b) *тип атрибута* (*attribute type*);

c) *значение атрибута* (*attribute value*);

d) *проверка значения атрибута* (*attribute value assertion*);

e) *контекст* (*context*);

f) *тип контекста* (*context type*);

g) *значение контекста* (*context value*);

h) *операционный атрибут* (*operational attribute*);

i) *атрибут пользователя* (*user attribute*);

j) *правило сопоставления* (*matching rule*).

3.5 Определения терминов по именам

Следующие термины определены в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2:

a) *псевдоним, имя псевдонима* (*alias, alias name*);

b) *выделенное имя* (*distinguished name*);

c) *имя (в Справочнике)* (*(directory) name*);

d) *потенциальное (предполагаемое) имя* (*purported name*);

e) *относительно выделенное имя* (*relative distinguished name*).

3.6 Определения терминов по распределенным операциям

Следующие термины определены в Рекомендации МСЭ-Т X.518 | ИСО/МЭК 9594-4:

a) *связанный DSA* (*bound DSA*);

b) *сцепление* (*chaining*);

c) *первоначальный исполнитель* (*initial performer*)

d) *отсылка* (*referral*).

3.7 Определения терминов по абстрактным службам

В настоящей Рекомендации | Международном стандарте применяются следующие определения терминов.

3.7.1 дополнительный поиск (additional search): Поиск, который начинается с **joinBaseObject**, определенного инициатором в запросе на операцию **search** (поиск).

3.7.2 содействующий член (contributing member): Член семейства в рамках составной статьи, который внес вклад в операцию Чтение, Поиск или Модификация статьи.

3.7.3 явно немаркированная статья (explicitly unmarked entry): Статья или член семейства, которые исключены из **SearchResult** согласно спецификации, заданной в атрибуте управления, который определяется управляющим правилом поиска.

3.7.4 группировка семейства (family grouping): Набор членов составного атрибута, которые сгруппированы вместе с целью проведения операции.

3.7.5 фильтр (filter): Проверка наличия или значения определенных атрибутов статьи для сужения области поиска.

3.7.6 инициатор (originator): Пользователь, который инициировал операцию.

3.7.7 участвующий член (participation member): Член семейства, который является либо содействующим членом, либо членом группировки семейства, который в целом соответствует фильтру операции **search**.

3.7.8 первичный поиск (primary search): Поиск, который начинается с **baseObject**, определенного инициатором в запросе на операцию поиска.

3.7.9 ослабление (relaxation): Последовательное изменение свойств фильтра в процессе поисковой операции для нахождения большего количества совпадающих статей, если их найдено немного, или меньшего количества совпадающих статей, если их найдено слишком много.

3.7.10 параметры управления службой (service controls): Параметры, которые передаются как часть операции и которые ограничивают различные аспекты ее рабочих характеристик.

3.7.11 трасса (strand): Группировка семейства, охватывающая всех членов на пути от листа (концевого члена семейства) до порождающего элемента включительно. Член семейства будет находиться в стольких трассах, сколько имеется концевых членов семейства ниже него (непосредственно или не непосредственно подчиненных).

3.7.12 результат, направленный в потоке (streamed result): тот или иной результат одной операции, включенный в несколько ответов.

4 Сокращения

В данной Рекомендации | Международном стандарте применяются следующие сокращения.

AVA Проверка значения атрибута

DIB Информационная база Справочника

DIT Информационное дерево Справочника

DMD Область управления Справочником

DSA Системный агент Справочника

DUA Агент пользователя Справочника

RDN Относительно выделенное имя

5 Соглашения по терминологии

За небольшими исключениями, эта спецификация Справочника была подготовлена в соответствии с *правилами представления общего текста МСЭ-Т | ИСО/МЭК*, ноябрь 2001 г.

Термин "спецификация Справочника" (как и "эта спецификация Справочника") означает Рекомендацию МСЭ-Т X.511| ИСО/МЭК 9594-3. Термин "спецификации Справочника" должен означать Рекомендации серии X.500 и все части Стандарта ИСО/МЭК 9594.

В данной спецификации Справочника используется термин "системы первого издания" для указания на системы, соответствующие первому изданию спецификаций Справочника, т. е. изданию 1988 года Рекомендаций МККТТ серии X.500 и изданию Стандарта ИСО/МЭК 9594:1990. В этой спецификации Справочника используется термин "системы второго издания" для указания на системы, соответствующие второму изданию спецификаций Справочника, т. е. изданию 1993 года Рекомендаций МСЭ-Т серии X.500 и изданию Стандарта ИСО/МЭК 9594:1995. В этой спецификации Справочника используется термин "системы третьего издания" для указания на системы, соответствующие третьему изданию спецификаций Справочника, т. е. изданию 1997 года Рекомендаций МСЭ-Т серии X.500 и изданию Стандарта ИСО/МЭК 9594:1998. В этой спецификации Справочника используется термин "системы четвертого издания" для указания на системы, соответствующие четвертому изданию спецификаций Справочника, т. е. изданиям 2001 года Рекомендаций МСЭ-Т X.500, X.501, X.511, X.518, X.519, X.520, X.521, X.525 и X.530, изданию 2000 года Рекомендации МСЭ-Т X.509 и частям 1–10 издания Стандарта ИСО/МЭК 9594:2001.

В настоящей спецификации используется термин *системы пятого издания* для ссылки на системы, соответствующие пятому изданию спецификаций Справочника, т. е. изданий 2005 года Рекомендаций МСЭ-Т X.500, X.501, X.509, X.511, X.518, X.519, X.520, X.521, X.525 и X.530 и частей 1–10 издания Стандарта ИСО/МЭК 9594:2005.

В данной спецификации Справочника нотация на языке ASN.1 дается полужирным шрифтом Helvetica. Когда типы и значения ASN.1 приводятся в обычном тексте, они выделяются полужирным шрифтом Helvetica, размером 9 пунктов. Названия процедур, упоминаемых при определении семантики обработки, выделяются в тексте полужирным шрифтом Times. Разрешения на управление доступом представляются курсивом шрифта Times.

Если элементы в списке пронумерованы (либо против них указаны "–" или буквы), то эти пункты должны рассматриваться как шаги в процедуре.

6 Обзор служб Справочника

Как описано в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2, службы Справочника обеспечиваются через пункты доступа к агентам пользователя Справочника (DUA), каждый из которых действует от имени пользователя. Эти понятия изображены на рис. 1. Через пункт доступа Справочник обеспечивает обслуживание своих пользователей посредством ряда операций Справочника.

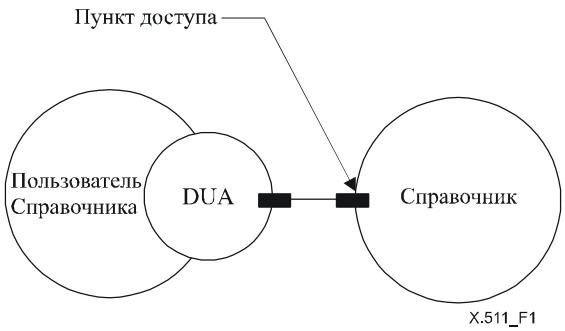

Рисунок 1 – Доступ к Справочнику

Существуют три различных класса операций Справочника:

a) операции Чтение из Справочника, которые запрашивают одну статью Справочника;

b) операции Поиск в Справочнике, которые имеют возможность запрашивать несколько статей Справочника; и

c) операции Модификация Справочника.

Операции Чтение из Справочника, операции Поиск в Справочнике и операции Модификация Справочника определены в разделах 9, 10 и 11, соответственно. Соответствие с операциями Справочника определено в Рекомендации МСЭ-Т X.519 | ИСО/МЭК 9594-5.

7 Типы информации и общие процедуры

7.1 Введение

В данном разделе указываются, а в некоторых случаях и определяются типы информации, которые впоследствии используются при определении операций Справочника. Рассматриваемые типы информации являются либо общими для нескольких операций, либо, вероятно, будут таковыми в будущем, либо являются достаточно сложными или независимыми, и поэтому заслуживают определения отдельно от операции, которая их использует.

Некоторые типы информации, используемые в определениях служб Справочника, фактически определены в других местах. В подразделе 7.2 перечислены эти типы с указанием источников их определений. В каждом из последующих подразделов (7.3–7.10) указывается и определяется один из типов информации.

В данном разделе также описываются общие элементы процедур, которые применяются для большинства или для всех операций Справочника.

7.2 Типы информации, определенные в других местах

Следующие типы информации определены в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2:

a) **Attribute** (атрибут);

b) **AttributeType** (тип атрибута);

c) **AttributeValue** (значение атрибута);

d) **AttributeValueAssertion** (проверка значения атрибута);

e) **Context** (контекст);

f) **ContextAssertion** (проверка контекста);

g) **DistinguishedName** (выделенное имя);

h) **Name** (Имя);

i) **OPTIONALLY-PROTECTED** (необязательная-защита);

j) **OPTIONALLY-PROTECTED-SEQ** (последовательность-необязательной-защиты);

k) **RelativeDistinguishedName** (относительно выделенное имя).

Следующий тип информации определен в Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6:

a) **PresentationAddress** (адрес в уровне представления).

Следующий тип информации определены в Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8:

a) **Certificate** (сертификат);

b) **SIGNED** (подписанный);

c) **CertificationPath** (траектория сертификации).

Следующий тип информации определен в Рекомендации МСЭ-Т X.880 | ИСО/МЭК 13712-1:

a) **InvokeId** (идентификатор вызова).

Следующие типы информации определены в Рекомендации МСЭ-Т X.518 | ИСО/МЭК 9594-4:

a) **OperationProgress** (продвижение операции);

b) **ContinuationReference** (ссылка на продолжение).

7.3 Общие аргументы

Информация **CommonArguments** (общие аргументы) может быть представлена для определения вызова каждой операции, которую Справочник может выполнить.

```
CommonArguments ::= SET {
        serviceControls          [30]  ServiceControls DEFAULT { },
        securityParameters       [29]  SecurityParameters OPTIONAL,
        requestor                [28]  DistinguishedName OPTIONAL,
        operationProgress        [27]  OperationProgress
                                       DEFAULT { nameResolutionPhase notStarted },
        aliasedRDNs              [26]  INTEGER OPTIONAL,
        criticalExtensions       [25]  BIT STRING OPTIONAL,
        referenceType            [24]  ReferenceType OPTIONAL,
        entryOnly                [23]  BOOLEAN DEFAULT TRUE,
        nameResolveOnMaster      [21]  BOOLEAN DEFAULT FALSE,
        operationContexts        [20]  ContextSelection OPTIONAL,
        familyGrouping           [19]  FamilyGrouping DEFAULT entryOnly }
```

Компонент **ServiceControls** определяется в п. 7.5. Его отсутствие рассматривается как эквивалент пустого множества параметров управления.

Компонент **SecurityParameters** определяется в п. 7.10. Если аргумент операции должен быть подписан инициатором, то компонент **SecurityParameters** должен быть включен в этот аргумент. Отсутствие **SecurityParameters** рассматривается как эквивалент пустого множества.

Выделенное имя **requestor** определяет инициатора конкретной операции. Оно содержит имя пользователя, указанное во время привязывания к Справочнику. Оно может потребоваться, когда запрос должен быть подписан (см. п. 7.10), и будет содержать имя пользователя, начавшего запрос.

ПРИМЕЧАНИЕ 1. – Если пользователь имеет несколько выделенных имен, отличающихся контекстом, то имя, используемое как значение **requestor**, должно быть первичным выделенным именем из известных. В противном случае аутентификация и управление доступом, основанные на значении **requestor**, не смогут работать должным образом.

Компоненты **operationProgress**, **referenceType**, **entryOnly**, **exclusions** и **nameResolveOnMaster** определены в Рекомендации МСЭ-Т X.518 | ИСО/МЭК 9594-4. Они выдаются из DUA в случаях:

a) когда действует ссылка на продолжение, выданная DSA в ответ на более раннюю операцию, а их значения скопированы в DUA из этой ссылки на продолжение; или

b) когда DUA представляет административного пользователя, который управляет Информационным деревом DSA, а в параметрах управления установлена опция **manageDSAIT**.

Компонент **aliasedRDNs** указывает для DSA, что компонент **object** операции был создан переименованием псевдонима во время более ранней попытки выполнения операции. Целочисленное значение указывает количество RDN в имени, которые получаются при переименовании псевдонима. (Значение должно быть установлено в ответе-отсылке из предыдущей операции.)

ПРИМЕЧАНИЕ 2. – Данный компонент предусмотрен для совместимости с реализациями Справочника первого издания. Агенты DUA (и DSA), реализованные согласно более поздним изданиям спецификаций Справочника, всегда будут опускать этот параметр в **CommonArguments** при последующем запросе. Таким образом, Справочник не будет сообщать об ошибке, если псевдонимы переименовываются в последующие псевдонимы.

Компонент **operationContexts** предоставляет набор контекстных проверок, применяемых для проверок значения атрибута и для выбора информации статьи, проводимых в рамках этой операции, которые, в противном случае, не содержат контекстных проверок для одинаковых типа атрибута и типа контекста. Если **operationContexts** отсутствует или не указывает конкретный тип атрибута или тип контекста, то DSA должен применять безусловные ("по умолчанию") контекстные проверки, как это описано ниже в п. 7.6.1, а также в п. 8.9.2.2 и п. 12.8 Рекомендации МСЭ-Т

X.501 | ИСО/МЭК 9594-2. Если выбран компонент **allContexts**, то все контексты для всех типов атрибута допустимы, а безусловные контекстные значения, которые могли бы быть выданы из DSA, не учитываются. (**ContextSelection** определен в п. 7.6.)

Компонент **familyGrouping** используется, чтобы указать на элементы семейства, которые должны быть выбраны для обработки в данной операции. Он описывается более полно в п. 7.3.2.

7.3.1 Критические расширения

Компонент **criticalExtensions** обеспечивает механизм внесения в список какого-либо набора расширений, являющихся критическими для рабочих характеристик операций, выполняемых Справочником. Если инициатор расширенной операции желает указать, что операция должна выполняться с одним или с несколькими расширениями (т. е., что выполнение операции без этих расширений не приемлемо), то он делает это посредством установки бита(ов) **criticalExtensions**, который(е) соответствует(ют) расширению(ям). Если Справочник (или некоторая его часть) не способен выполнить критическое расширение, то он выдает уведомление **unavailableCriticalExtension** (так же, как **serviceError** или **PartialOutcomeQualifier**). Если Справочник не способен выполнить расширение, которое не является критическим, то он игнорирует наличие такого расширения.

Настоящая спецификация Справочника не устанавливает правил в отношении порядка, в котором исполняющий DSA осуществляет декодирование и обработку блоков PDU, которые он принимает. Агент DSA, который принимает неизвестное критическое расширение, возвращает **ServiceError** с проблемой **unavailableCriticalExtension** для сообщения о невыполнении операции.

Эти спецификации Справочника определяют ряд расширений. Расширения принимают такие формы, как дополнительные пронумерованные биты в BIT STRING или как дополнительные компоненты в SET или в SEQUENCE; они игнорируются системами первого издания. Каждое такое расширение снабжается целочисленным идентификатором, являющимся числом битов, которое может быть установлено в компоненте **criticalExtensions**. Если критичность расширения определена как "критическая", то DUA должен установить соответствующий бит в **criticalExtensions**. Если установленная критичность является "некритической", то DUA может устанавливать или не устанавливать соответствующий бит в **criticalExtensions**.

Расширения, их идентификаторы, операции, для которых они разрешаются, рекомендуемая критичность, пункты, в которых они определяются, и соответствующие параметры управления LDAP показаны в табл. 1.

Таблица 1 – Расширения

Расширение	Идентификатор	Операции	Критичность	Определено (подпункты)	Параметр управления LDAP
subentries (подстатьи)	1	Все	Некритическая	17.5	1.3.6.1.4.1.4203.1.10.1
copyShallDo (выполнить копию)	2	Чтение, Сравнение, Список, Поиск	Некритическая	17.5	
attribute size limit (ограничение размера атрибута)	3	Чтение, Поиск	Некритическая	17.5	
extraAttributes (дополнительные атрибуты)	4	Чтение, Поиск	Некритическая	17.6	
ModifyRightsRequest (запрос права модификации)	5	Чтение	Некритическая	19.1	
pagedResultsRequest (запрос результатов в виде страниц)	6	Список, Поиск	Некритическая	10.1	1.2.840.113556.1.4.319
matchedValuesOnly (только совпадающие значения)	7	Поиск	Некритическая	10.2	1.2.826.0.1.3344810.2.3
extendedFilter (расширенный фильтр)	8	Поиск	Некритическая	10.2	
targetSystem (целевая система)	9	Добавление статьи	Критическая	11.1	
useAliasOnUpdate (использование псевдонима при обновлении)	10	Добавление статьи, Удаление статьи, Модификация статьи	Критическая	11.1	
newSuperior (новый предшествующий элемент)	11	Модификация выделенного имени	Критическая	11.4	
manageDSAIT (управление информационным деревом DSA)	12	Все	Критическая	7.5, 7.13	2.16.840.1.113730.3.4.2
useContexts (использование контекста)	13	Чтение, Сравнение, Список, Поиск, Добавление статьи, Модификация статьи, Модификация выделенного имени	Некритическая	7.6, 7.8	

Таблица 1 – Расширения

Расширение	Идентификатор	Операции	Критичность	Определено (подпункты)	Параметр управления LDAP
partialNameResolution (частичная проверка имени)	14	Чтение, Поиск	Некритическая	7.5	
overspecFilter (переопределение фильтра)	15	Поиск	Некритическая	10.1.3 f)	
selectionOnModify (выбор при модификации)	16	Модификация статьи	Некритическая	11.3.2	
Security parameters – Response (Параметры безопасности – Ответ)	17	Все	Некритическая	7.10	
Security parameters – Operation code (Параметры безопасности – Код операции)	18	Все	Некритическая	7.10	
Security parameters – Attribute certification path (Параметры безопасности – Траектория сертификации атрибута)	19	Все	Некритическая	7.10	
Security parameters – Error Protection (Параметры безопасности – Ошибка защиты)	20	Все	Некритическая	7.10	
SPKM Credentials (Удостоверения SPKM)	21	Привязывание к Справочнику	(Примечание 3)	8.1.1	
Bind token – Response (Маркер привязывания – Ответ)	22	Привязывание к Справочнику	Некритическая	8.1.1	
Bind token (Маркер привязывания) – Bind Int. Alg, Bind Int Key, Conf Alg и Conf Key Info	23	Привязывание к Справочнику	Некритическая	8.1.1	
Bind token (Маркер привязывания) – DIRQOP (устаревший)	24	Привязывание к Справочнику	Некритическая	8.1.1	
Service administration (Администрирование службы)	25	Чтение, Поиск, Модификация статьи	Критическая	10.2.2, 13, пункт 16 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2	
entryCount (количество статей)	26	Поиск	Некритическая	10.1.3	
hierarchySelection (выбор иерархии)	27	Поиск	Некритическая	7.5	
relaxation (ослабление)	28	Поиск	Некритическая	7.8	
familyGrouping (группировка семейства)	29	Сравнение, Поиск, Удаление статьи	Некритическая Некритическая Критическая	7.3.2, 7.8.3 и 9.2.2 10.2 11.2.2	
familyReturn (выдача семейства)	30	Чтение, Поиск, Модификация статьи	Некритическая Некритическая Некритическая	7.6.4, 7.7.1 и 9.1.3 10.2.3 11.3.3	
dnAttributes (атрибуты выделенного имени)	31	Поиск	Некритическая	10.2.2	
friend attributes (дружественные атрибуты)	32	Чтение, Поиск	Некритическая	7.6, 7.8.2	
Abandon of paged results (отмена результатов в виде страниц)	33	Список, Поиск	Критическая	7.9	
Paged results on the DSP (результаты в видне страниц на DSP)	34	Список, Поиск	Некритическая	7.9	
replaceValues (замененные значения)	35	Модификация статьи	Критическая	11.3.1, 11.3.2	

ПРИМЕЧАНИЕ 1. – Первому расширению присваивают идентификатор 1, что соответствует биту 1 в BIT STRING. Бит 0 в BIT STRING не используется.

ПРИМЕЧАНИЕ 2. – Использование зашифрованного или подписанного и зашифрованного преобразования для безопасности, либо любой защиты уведомлений об ошибке или результатов в операциях Добавление статьи, Удаление статьи, Модификация статьи, Модификация DN требует протокола версии 2 или выше.

ПРИМЕЧАНИЕ 3. – Расширение удостоверений SPKM должно быть критическим, за исключением случаев, когда оно используется в установленных ассоциациях, использующих версию 2 или выше.

7.3.2 Группировка семейства

Группировка семейства позволяет одиночному члену семейства, нескольким членам семейства или всем членам семейства составной статьи сгруппироваться вместе для объединенного рассмотрения до проведения операции. Эта семантика может применяться к следующим операциям (как показано в описаниях ниже): Сравнение (чтобы определить область применения, в пределах которой сравниваемый атрибут мог бы находиться), Поиск (чтобы определить группировки, для которых фильтрация могла бы иметь место), Удаление статьи (чтобы определить группировки для удаления). Для выбора членов семейства используется следующая запись на языке ASN.1:

```
FamilyGrouping ::= ENUMERATED {
    entryOnly       (1),
    compoundEntry   (2),
    strands         (3),
    multiStrand     (4) }
```

Параметр **entryOnly** означает, что конкретный член семейства, выбранный операцией, должен рассматриваться в группе. Это – значение "по умолчанию", что гарантирует совместимость вниз с предыдущими изданиями спецификаций Справочника.

Параметр **compoundEntry** означает, что целая составная статья, выбранная операцией, должна рассматриваться как модуль с объединением всех атрибутов. Для операций Удаление статьи это применимо только тогда, когда указанное имя объекта является порождающим элементом для составной статьи, что вызывает удаление всех членов семейства той же самой операцией (которая подчиняется управлению доступом).

Параметр **strands** означает, что все трассы, связанные с членом семейства, должны быть выбраны операцией. Эта опция не допустима для операции Удаление статьи. Для операции Поиск отдельные трассы рассматриваются для целей фильтрации. Если объединенный набор атрибутов одной или нескольких трасс соответствует фильтру, то считается, что составная статья соответствует фильтру. Если базовый объект – это порожденный член, то рассматриваются только те трассы, которые проходят через этот базовый объект. Для операций Сравнение все атрибуты всех членов семейства во всех трассах, к которым принадлежит статья, должны использоваться для сравнения.

Параметр **multiStrand** применим только к операции Поиск и определяет правило сопоставления для фильтрации информации семейства. Он игнорируется для других операций. Он определяет, что одна трасса от каждого семейства в пределах составной статьи должна рассматриваться один раз, но во всех комбинациях. Параметр **multiStrand** не применим, когда базовый объект – это порожденный член семейства, в этом случае параметр **multiStrand** игнорируется и подставляется параметр **entryOnly**.

7.4 Общие результаты

Информация **CommonResults** или **CommonResultsSeq** должна присутствовать для определения результата каждой операции обращения, которую Справочник может выполнить. Кроме того, она присутствует в любом выдаваемом уведомлении об ошибке.

```
CommonResults ::= SET {
    securityParameters      [30]    SecurityParameters                  OPTIONAL,
    performe                [29]    DistinguishedName                   OPTIONAL,
    aliasDereferenced       [28]    BOOLEAN                             DEFAULT FALSE,
    notification            [27]    SEQUENCE SIZE (1 ..MAX) OF Attribute OPTIONAL }

CommonResultsSeq ::= SEQUENCE {
    securityParameters      [30]    SecurityParameters                  OPTIONAL,
    performer               [29]    DistinguishedName                   OPTIONAL,
    aliasDereferenced       [28]    BOOLEAN                             DEFAULT FALSE,
    notification            [27]    SEQUENCE SIZE (1 ..MAX) OF Attribute OPTIONAL }
```

ПРИМЕЧАНИЕ. – Параметры **CommonResults** и **CommonResultsSeq** состоят из одинаковых компонентов. Первый используется, когда он включен во множество типов посредством типа **COMPONENT OF**, а последний используется аналогично в типах "последовательность".

Компонент **SecurityParameters** определяется в п. 7.10. Если результат должен быть подписан Справочником, то компонент **SecurityParameters** должен включаться в результат. Отсутствие компонента **SecurityParameters** рассматривается как эквивалент пустого множества.

Выделенное имя компонента **performer** идентифицирует исполнителя конкретной операции. Он может потребоваться, когда результат должен быть подписан (см. п. 7.10), и должен содержать имя того DSA, который подписал результат.

Компонент **aliasDereferenced** устанавливается в состояние **TRUE** (ИСТИНА), когда потенциальное (предполагаемое) имя какого-либо объекта или базового объекта, который является целью операции, содержит псевдонимы, которые были переименованы.

Компонент **notification** должен использоваться для определения выдаваемого результата и ошибки APDU (Application Protocol Data Unit, протокольный блок данных прикладного уровня), например, обеспечивая более точные сведения об ошибке. Стандартные атрибуты уведомления определены в п. 5.12 Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6. Такие атрибуты уведомления не обязательно хранятся в статьях Справочника.

7.5 Параметры управления службой

Параметр **ServiceControls** содержит параметры управления, если они имеются, которые направляют или ограничивают обеспечение службы.

```
ServiceControls ::= SET {
        options                 [0] ServiceControlOptions DEFAULT { },
        priority                [1]   INTEGER { low (0), medium (1), high (2)}  DEFAULT medium,
        timeLimit               [2]   INTEGER OPTIONAL,
        sizeLimit               [3]   INTEGER OPTIONAL,
        scopeOfReferral         [4]   INTEGER { dmd(0), country(1) }  OPTIONAL,
        attributeSizeLimit      [5]   INTEGER OPTIONAL,
        manageDSAITPlaneRef     [6]   SEQUENCE {
                dsaName                       Name,
                agreementID                   AgreementID } OPTIONAL,
        serviceType             [7]   OBJECT IDENTIFIER OPTIONAL,
        userClass               [8]   INTEGER OPTIONAL}

ServiceControlOptions ::= BIT STRING {
        preferChaining                  (0),
        chainingProhibited              (1),
        localScope                      (2),
        dontUseCopy                     (3),
        dontDereferenceAliases          (4),
        subentries                      (5),
        copyShallDo                     (6),
        partialNameResolution           (7),
        manageDSAIT                     (8),
        noSubtypeMatch                  (9),
        noSubtypeSelection              (10),
        countFamily                     (11),
        dontSelectFriends               (12),
        dontMatchFriends                (13),
        allowWriteableCopy              (14) }
```

Компонент **options** содержит несколько указателей, каждый из которых, если он установлен, определяет предложенное условие. Таким образом:

a) **preferChaining** указывает, что при обеспечении службы предпочтение должно отдаваться сцеплению, а не отсылкам. Справочник не обязан следовать этому предпочтению.

b) **chainingProhibited** указывает, что сцепление и другие методы распределения запроса по Справочнику, запрещены.

c) **localScope** указывает, что операция должна быть ограничена местной областью применения. Определение этой опции само является местным вопросом, например, в пределах одного DSA или одной DMD.

d) **dontUseCopy** указывает, что скопированная информация, определенная в Рекомендации МСЭ-Т X.518 | ИСО/МЭК 9594-4, не должна использоваться при обеспечении службы.

e) **dontDereferenceAliases** указывает, что любой псевдоним, используемый для идентификации статьи, затронутой операцией, не должен переименовываться.

   ПРИМЕЧАНИЕ 1. – Это необходимо для того, чтобы разрешить ссылку непосредственно на статью псевдонима, а не на ту, на которую указывает статья псевдонима, например, для чтения статьи псевдонима.

f) **subentries** указывает, что для операции Поиск или Список доступны только подстатьи; обычные статьи становятся недоступными, т. е. Справочник ведет себя так, как будто обычные статьи не существуют. Если не установлен этот параметр управления службой, то операция обращается только к обычным статьям, а подстатьи станут недоступными. Этот параметр управления службой игнорируется для других операций, не являющихся операцией Поиск или Список.

   ПРИМЕЧАНИЕ 2. – Тем не менее наблюдается влияние подстатей на управление доступом, на схему, на совокупность атрибутов, даже если подстатьи недоступны.

   ПРИМЕЧАНИЕ 3. – Если этот параметр управления службой установлен, то обычные статьи все же могут указываться в качестве базового объекта операции.

g) **copyShallDo** указывает, что если Справочник способен только частично, а не полностью удовлетворить запрос на копию статьи, то он не должен сцеплять запрос. Это действительно только в случае, когда **dontUseCopy** не установлен. Если **copyShallDo** не установлен, то Справочник будет использовать скопированные данные только в случае, когда они достаточно полны, чтобы разрешить завершение операции полностью в виде копии. Запрос может быть удовлетворен только частично из-за того, что некоторые из требуемых атрибутов отсутствуют в скопированной копии, или потому, что некоторые из значений атрибутов для данного атрибута отсутствуют в скопированной копии, или потому, что DSA не содержит всю контекстную информацию для значений атрибутов, которые он содержит, или потому, что DSA, содержащий скопированные данные, не поддерживает требуемые правила сопоставления данных. Если **copyShallDo** установлен, но Справочник не способен полностью удовлетворить запрос, то будет установлен параметр **incompleteEntry** в выдаваемой информации статьи.

h) **partialNameResolution** указывает, что если Справочник способен проверить только часть потенциального имени в операции Чтение или Поиск, т. е. он собирается выдать **nameError**, то статья, чье имя состоит из всех проверенных RDN, должна рассматриваться как цель операции, а параметр **partialName** в выдаваемом результате устанавливается в **TRUE**. Этот параметр управления службой игнорируется для других операций, не являющихся операцией Чтение или Поиск.

ПРИМЕЧАНИЕ 4. – Если этот параметр управления службой установлен, то потенциальное имя – это контекстная префиксная статья, доступ к которой отклонен, а инициатор имеет доступ к старшей статье, и тогда существование контекстной префиксной статьи будет косвенно раскрыто инициатору, даже если разрешение *DiscloseOnError* (раскрытие в уведомлении об ошибке) для этой статьи отклонено.

i) **manageDSAIT** указывает, что операция была запрошена административным пользователем для управления Информационным деревом DSA. Если в DSA существуют несколько уровней репликации, которыми нужно управлять, а параметр управления службой **manageDSAITPlaneRef** не был включен в операцию, то DSA выбирает подходящий уровень репликации для этой операции.

j) **noSubtypeMatch** указывает, что сопоставление подтипа атрибута не должно проводиться. Этот параметр управления службой игнорируется для других операций, не являющихся операциями Сравнение или Поиск.

k) **noSubtypeSelection** указывает, что выбор подтипа не должен проводиться.

l) **countFamily** указывает, что каждый член составной статьи должен быть подсчитан как отдельная статья, например, с целью определения размера и административных пределов, а также для управления ослаблением. Если этот параметр управления не установлен, то члены составного атрибута должны считаться одиночной статьей.

m) **dontSelectFriends** указывает на то, что спецификация атрибута привязки в выборке информации о статье не включает автоматически атрибутов "друзей" в этой выборке.

n) **dontMatchFriends** указывает на то, что спецификация атрибута привязки в элементе фильтра может быть заполнена только значениями атрибута привязки, а не атрибутами "друзей".

o) **allowWriteableCopy** указывает на то, что DSE типа **writeableCopy** приемлем при предоставлении требования на услугу запроса.

ПРИМЕЧАНИЕ 5. – Управление услугой **allowWriteableCopy** отличается от **copyShallDo** тем, что оно используется для указания того, что затребована полная копия, но что она не должна быть основным экземпляром, тогда как **copyShallDo** используется для указания того, что приемлема любая, полная или неполная, копия.

Если компонент **options** опущен, то предполагается следующее: сцеплению не оказывается предпочтение, но сцепление не запрещено; нет ограничения на область применения операции; использование копии разрешено; псевдонимы должны переименовываться (за исключением операций модификации, для которых переименование псевдонима не поддерживается); подстатьи недоступны; операции, не полностью удовлетворяющие условиям использования скопированных данных, подвергаются дальнейшему сцеплению. Однако эти правила "по умолчанию" могут быть заменены правилами поиска в пределах административных областей, зависящих от службы.

Компонент **priority** указывает приоритет, с которым должна обеспечиваться служба; он имеет значения: низкий, средний или высокий. Следует отметить, что эта услуга не является гарантированной, поскольку Справочник, как единое целое, не обеспечивает организацию очереди. Между этим компонентом и использованием приоритетов в нижележащих уровнях нет взаимосвязи.

Компонент **timeLimit** указывает максимальный промежуток времени в секундах, в пределах которого служба должна быть предоставлена. Если это ограничение не может быть удовлетворено, то выдается уведомление об ошибке. Если этот компонент опущен, то ограничение времени не устанавливается. В случае превышения лимита времени при операции Список или Поиск результатом будет произвольная выборка полученных к этому моменту результатов.

ПРИМЕЧАНИЕ 6. – В этот компонент не входит отрезок времени, затраченный на обработку запроса в течение прошедшего времени: любое число DSA может участвовать в обработке запроса в течение прошедшего времени.

Компонент **sizeLimit** применяется только к операциям Список и Поиск. Он указывает на максимальное число выдаваемых статей, если результаты в виде страниц не будут выданы в ответе. В случае превышения ограничения объема результаты операций Список или Поиск могут быть произвольной выборкой накопленных результатов, численно равной этому пределу объема. Любые дальнейшие результаты будут аннулированы. Если результаты в виде страниц выдаются в ответе, значение sizeLimit (предел объема) игнорируется агентом DSA, осуществляющего вызов страниц, как подробно описано в п. 7.9.

Компонент **scopeOfReferral** указывает область, к которой будет иметь отношение отсылка, выданная из DSA. В зависимости от выбранного значения (**dmd** или **country**) будут выданы только отсылки к другим DSA в пределах выбранной области. Это применимо к отсылкам как в уведомлении об ошибке **referral**, так и в параметре **unexplored** в результатах операций Список и Поиск.

Компонент **attributeSizeLimit** указывает наибольший размер любого атрибута (т. е. тип и все его значения), который включается в выдаваемую информацию статьи. Если атрибут превышает это ограничение, то все его значения в выдаваемой информации статьи опускаются, а в этой выдаваемой информации статьи устанавливается параметр **incompleteEntry**. Размер атрибута означает его размер в октетах в местном конкретном синтаксисе того DSA, который содержит эти данные. Из-за различий способов хранения данных прикладными программами это ограничение является неточным. Если этот параметр не указан, то никакого ограничения не подразумевается.

ПРИМЕЧАНИЕ 7. – Значения атрибута, выдаваемые как часть Выделенного имени статьи, освобождаются от этого ограничения.

Некоторые комбинации **priority**, **timeLimit** и **sizeLimit** могут приводить к конфликтам. Например, короткий промежуток времени может находиться в конфликте с низким приоритетом; большой лимит размера может находиться в конфликте с коротким лимитом времени и т. д.

Компонент **manageDSAITPlaneRef** указывает, что операция запрошена административным пользователем для управления конкретным уровнем репликации Информационного дерева DSA. Параметр управления службой **manageDSAITPlaneRef** игнорируется, если опция **manageDSAIT** не установлена. Уровень указывается компонентом **dsaName**, который является именем обеспечивающего DSA, и компонентом **agreementID**, который содержит идентификатор соглашения о копировании.

Параметр управления службой **serviceType** действует только при запросе **search**, который запускает свою начальную фазу выполнения в пределах зависящей об службы административной области; в других случаях он игнорируется. Если он установлен, то это увеличивает вероятность получения полезной информации уведомления, выдаваемой в случае неверно сформулированного запроса **search**.

Параметр управления службой **userClass** действует только при запросе **search**, который запускает свою начальную фазу выполнения в пределах зависящей от службы административной области; в других случаях он игнорируется. Он указывает класс пользователя. Он позволяет инициатору определить другой класс пользователя, а не тот, который иначе применил бы Справочник. Если он установлен, то это увеличивает также вероятность получения полезной информации уведомления, выдаваемой в случае неверно сформулированного запроса **search**.

7.6 Выбор информации из статьи

Параметр **EntryInformationSelection** указывает, какая информация запрашивается из статьи в службе с поиском.

```
EntryInformationSelection ::= SET {
    attributes                  CHOICE {
        allUserAttributes           [0]   NULL,
        select                      [1]   SET OF AttributeType
        -- пустое множество означает, что никакие атрибуты не требуются --  } DEFAULT allUserAttributes : NULL,
    infoTypes                   [2]         INTEGER {
        attributeTypesOnly              (0),
        attributeTypesAndValues         (1)}  DEFAULT attributeTypesAndValues,
    extraAttributes             CHOICE {
        allOperationalAttributes    [3]   NULL,
        select                      [4]   SET SIZE (1 ..MAX) OF AttributeType } OPTIONAL,
    contextSelection            ContextSelection OPTIONAL,
    returnContexts              BOOLEAN DEFAULT FALSE,
    familyReturn                FamilyReturn DEFAULT
                                { memberSelect contributingEntriesOnly } }

ContextSelection ::= CHOICE {
    allContexts         NULL,
    selectedContexts    SET SIZE (1 ..MAX) OF TypeAndContextAssertion }

TypeAndContextAssertion ::= SEQUENCE {
    type                AttributeType,
    contextAssertions   CHOICE {
        preference          SEQUENCE OF ContextAssertion,
        all                 SET OF ContextAssertion } }

FamilyReturn ::= SEQUENCE {
    MemberSelect    ENUMERATED {
        contributingEntriesOnly     (1),
        participatingEntriesOnly    (2),
        compoundEntry               (3) },
    familySelect    SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id  OPTIONAL }
```

Компонент **attributes** определяет атрибуты пользователя и атрибуты операции, о которых запрашивается информация:

a) Если выбрана опция **select**, то перечисляются необходимые атрибуты. Если список пуст, то никакие атрибуты не должны выдаваться. Если атрибут присутствует, то должна выдаваться информация о выбранном атрибуте. Если ни один из выбранных атрибутов не присутствует, то выдается только уведомление об ошибке **attributeError** с указанием проблемы (причины) **noSuchAttributeOrValue**.

b) Если выбрана опция **allUserAttributes**, то запрашивается информация обо всех атрибутах пользователя в статье.

Информация атрибута выдается только в случае, когда права доступа достаточны. Только в случае, когда права доступа запрещают чтение всех запрашиваемых значений атрибутов, будет выдаваться **securityError** (с указанием проблемы **insufficientAccessRights**). Следует отметить, что управление доступом также применяется к атрибутам и значениям, которые могут выдаваться согласно компонентам **EntryInformationSelection**, и это может дополнительно ограничивать объем выдаваемой информации.

ПРИМЕЧАНИЕ 1. – Управление доступом также применяется к атрибутам и значениям, которые могут выдаваться согласно компонентам **EntryInformationSelection**, и это может дополнительно ограничивать объем выдаваемой информации.

Компонент **infoTypes** указывает, требуется ли информация о типе атрибута и информация о его значении ("по умолчанию"), либо требуется информация только о типе атрибута. Если тип атрибута включает в себя другие атрибуты, например, атрибут **family-information**, то это значение (значения) должно выдаваться независимо от установки компонента **infoTypes**, но спецификация **infoTypes** должна применяться к этим содержащимся атрибутам. Если компонент **attributes** установлен так, что не запрашиваются никакие атрибуты, то данный компонент не принимается во внимание.

Компонент **extraAttributes** указывает набор дополнительных атрибутов пользователя и атрибутов операции, о которых запрашивается информация. Если установлена опция **allOperationalAttributes**, то запрашивается информация обо всех атрибутах операций Справочника, содержащихся в статье. Если выбрана опция **select**, то запрашивается информация о перечисленных атрибутах.

ПРИМЕЧАНИЕ 2. – Данный компонент может использоваться для запроса информации, например, о конкретных атрибутах операций, если компонент **attributes** установлен в **allUserAttributes**, или обо всех атрибутах операций. Если один и тот же атрибут имеется в списке или применяется как в **attributes,** так и в **extraAttributes**, то он обрабатывается так, как будто он запрашивается только один раз.

Запрос конкретного атрибута всегда обрабатывается как запрос атрибута и всех *подтипов* этого атрибута (за исключением запросов, обрабатываемых системами первого издания), если опция параметра управления службой **noSubtypeSelection** не установлена. Если опция параметра управления **noSubtypeSelection** установлена, то выдаются только требуемые атрибуты, без их подтипов. Аналогично запрос конкретного атрибута, имеющего "друзей", обрабатывается как запрос атрибута и всех атрибутов "друзей", при условии что опция управления услугой **dontSelectFriends** не установлена.

В ответе на запрос об информации атрибута Справочник обрабатывает все *совокупные атрибуты* статьи так, как будто они были фактическими атрибутами пользователя в статье, т. е. они выбираются подобно другим атрибутам пользователя и объединяются в выдаваемой информации статьи. Запрос с параметром **allUserAttributes** запрашивает все совокупные атрибуты статьи, как и обычные атрибуты статьи. Атрибут является совокупным атрибутом статьи, если выполняется все нижеследующее:

a) он расположен в подстатье, спецификация поддерева которой содержит данную статью;

b) он не исключается при наличии в статье значения атрибута **collectiveExclusions**, равного этому типу совокупного атрибута; и

c) он допускается контекстным правилом для классов структурных объектов данной статьи.

Компонент **contextSelection** используется для указания, какие значения атрибутов должны выдаваться для атрибутов, выбранных компонентом **attributes** или **extraAttributes**. Компонент **contextSelection** применяется только к тем значениям атрибутов, которые являются кандидатами на выдачу в соответствии со значениями других компонентов параметра **EntryInformationSelection**. Обработка компонента **contextSelection** и использование операций "по умолчанию", если он отсутствует, обсуждаются в пп. 7.6.1–7.6.3.

Если компонент **infoTypes** имеет значение, не запрашивающее значений атрибутов, или компонент **attributes** не запрашивает никаких атрибутов, то компонент **contextSelection** не имеет значения. Если в результате применения **contextSelection** отсутствуют значения атрибута, намеченного для выдачи, то этот атрибут может быть выдан без значений.

Компонент **returnContexts** используется, чтобы запросить Справочник выдать значения атрибутов вместе с их связанными контекстными списками. Если этот компонент отсутствует или указан со значением **FALSE**, то никакая контекстная информация не выдается в ответе. Если этот компонент указан со значением **TRUE**, то вся контекстная информация выдается для каждого выдаваемого значения атрибута. Следует отметить, что компонент **contextSelection** не проводит отбор выдаваемой контекстной информации, когда **returnContexts** равно **TRUE**.

Компонент **familyReturn** (если присутствует) используется для определения того, какие статьи в рамках составной статьи должны быть выданы, если один или несколько членов семейства были маркированы (см. п. 7.6.4).

7.6.1 Использование contextSelection или его значений "по умолчанию"

Компонент **contextSelection** используется для выбора определенных значений атрибутов для тех атрибутов, которые указаны компонентом **attributes** или **extraAttributes**. Применяется **contextSelection** только к тем значениям атрибутов, которые являются кандидатами на выдачу в соответствии со значениями других компонентов параметра **EntryInformationSelection**. Для каждого значения атрибута любой **contextSelection**, управляющий типом его атрибута, должен быть установлен в **TRUE** (как определено в п. 7.6.2), чтобы это значение атрибута было выбрано.

Считается, что **contextSelection** управляет типом атрибута, если выполняется какое-либо из следующих условий:

– **ContextSelection** указывает **allContexts** (когда выбраны все значения атрибутов для всех типов атрибута);

– **ContextSelection** имеет значение **selectedContexts**, которое содержит **TypeAndContextAssertion**, чей тип совпадает с типом атрибута или с его супертипом; либо

– **ContextSelection** имеет значение **selectedContexts**, которое содержит **TypeAndContextAssertion**, чей тип – **id-oa-allAttributeTypes**.

Если **contextSelection** не задан или он не управляет заданным типом атрибута, то должно применяться значение **contextSelection** "по умолчанию". В дополнение к **contextSelection** в **EntryInformationSelection** имеются три потенциальных источника для **contextSelection** (выбора контекста): тот, который указывает операцию в целом; тот, который доступен в пределах подстатей в DIT, и тот, который доступен в местном DSA. Они применяются согласно следующим приоритетам:

1) Если **contextSelection** присутствует в **EntryInformationSelection** и управляет заданным типом атрибута, как описано выше, то он должен применяться.
2) Если **contextSelection** не присутствует в **EntryInformationSelection**, либо присутствует, но не управляет заданным типом атрибута, то должен применяться **operationContexts**, который был установлен для этой операции согласно п. 7.3, если он присутствует и управляет заданным типом атрибута, как описано выше.
3) Если запрос не имеет ни **contextSelection** в **EntryInformationSelection**, ни **operationContexts** для операции, или ни один из них не управляет заданным атрибутом, то значения атрибута **contextAssertionDefaults** в контекстных определениях подстатей (если они имеются), которые управляют статьей, должны применяться как **selectedContexts**. (Контекстные определения подстатьей описаны в п. 14.7 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2).
4) Если **contextSelection**, который управляет заданным типом атрибута, отсутствует от источников, описанных выше, то DSA может применить местное значение **contextSelection** "по умолчанию". Такое значение "по умолчанию" обычно должно отражать такие местные параметры, как язык, расположение места размещения DSA или текущее время дня, но DSA может применять его разными способами для разных DUA, за которые он отвечает.
5) Если **contextSelection**, который управляет заданным типом атрибута, доступен от одного из этих источников, то все значения атрибута считаются выбранными (т. е. **allContexts** принят в качестве основного значения "по умолчанию").

ПРИМЕЧАНИЕ. – Значение **contextSelection** "по умолчанию", которое управляет заданным типом атрибута и определяет некоторый контекстный тип, будет применяться в дополнение к предшествующему **contextSelection,** управляющему тем же самым типом атрибута, но определяющим другой контекстный тип в том же порядке приоритетов, который описан выше.

7.6.2 Вычисление contextSelection

Значение **contextSelection** равно TRUE (т. е. выбирает заданное значение атрибута), если:

a) **allContexts** задан (это разрешает контекстному выбору отменять любое значение "по умолчанию", которое иначе могло бы применяться, если бы **contextSelection** был опущен); или
b) каждый **TypeAndContextAssertion** в **selectedContexts** равен TRUE, как описано в п. 7.6.3.

В других случаях значение **contextSelection** равно FALSE.

7.6.3 Вычисление TypeAndContextAssertion

Значение **TypeAndContextAssertion** равно TRUE (т. е. выбирает заданное значение атрибута), если:

a) тип атрибута не совпадает с **type** в **TypeAndContextAssertion** (и не является его подтипом), а **type** в **TypeAndContextAssertion** не равно **id-oa-allAttributeTypes**. В этом случае **TypeAndContextAssertion** не применим к типу атрибута заданного значения атрибута и, таким образом, запрещает выбор такого значения атрибута; или
b) **contextAssertions** в **TypeAndContextAssertion** для значения атрибута равно TRUE, как определяется ниже.

ПРИМЕЧАНИЕ 1. – Компонент **id-oa-allAttributeTypes** со значением **OBJECT IDENTIFIER** может использоваться как значение компонента **type** в **TypeAndContextAssertion**, чтобы вынудить применение **contextAssertions** к значению атрибута для любого типа атрибута.

Компонент **contextAssertions** выражен как упорядоченная последовательность предпочтительных контекстов или как составное множество контекстных утверждений:

a) Если задан компонент **all**, то **contextAssertions** равен TRUE для любого значения атрибута только в случае, когда каждый параметр **ContextAssertion** в SET равен TRUE**,** как определено в 8.9.2.4 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.
b) Если задан компонент **preference**, то каждый параметр **ContextAssertion** в **SEQUENCE** применяется, в свою очередь, ко всем кандидатам значений атрибутов одного и того же типа атрибута, пока **ContextAssertion** не становится равным TRUE**,** как определено в 8.9.2.4 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. (Признак перехода в аварийный режим, если он имеется, не учитывается, пока вся SEQUENCE не завершится.) Как только **ContextAssertion** становится равным TRUE для одного кандидата значения атрибута, то он должен быть применен для каждого кандидата значения атрибута одного и того же типа, но следующий **ContextAssertion** в SEQUENCE игнорируется.

ПРИМЕЧАНИЕ 2. – Компонент **preference** обеспечивает средства для выбора, который будет определен в виде первого, второго и т. д. выборов контекста (например, Язык = французский, но если не французский, то Язык = английский).

В остальных случаях значение **TypeAndContextAssertion** равно FALSE.

7.6.4 Ответное сообщение семейства

Компонент **familyReturn** используется для определения того, какие статьи в рамках составной статьи должны быть выданы, если один или несколько членов семейства были маркированы как содействующие или участвующие члены. Процедуры маркировки членов семейства подробно описываются в п. 7.13.
Компонент **memberSelect** определяет, какие статьи выбраны для выдачи в результате:

- Значение **contributingEntriesOnly** указывает, что должны выдаваться только члены семейства, маркированные операцией как содействующие члены. При операции Чтение или Модификация статьи – это член семейства, указанный аргументом операции **object**; для операции Поиск включаются главы семейства, которые внесли вклад в соответствие.
- Значение **participatingEntriesOnly** указывает, что должны выдаваться только члены семейства, маркированные операцией как участвующие члены. При операции Чтение или Модификация статьи – это тот же самый член, как и в случае **contributingEntriesOnly**.
- Значение **compoundEntry** указывает, что должен выдаваться каждый член семейства в пределах составной статьи, кроме тех, которые, возможно, не были явно выделены управляющим правилом поиска при операции Поиск.

Компонент **memberSelect** дополняется компонентом **familySelect**, который определяет, что все порожденные члены выбранных семейств должны выдаваться в дополнение к тем, которые определяет **memberSelect**. Порядок следования членов не имеет значения. Семейство определяется структурным объектным классом членов семейства, непосредственно подчиненным порождающему элементу. Этот компонент не оказывает влияние, если компонент **memberSelect** имеет значение **compoundEntry**.

ПРИМЕЧАНИЕ. – Управляющее правило поиска может изменять перечень информации, которая должна выдаваться (см. п. 16.10 из Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2).

7.7 Информация статьи

7.7.1 Тип данных для информации статьи

Тип данных **EntryInformation** определяет передаваемую информацию, выбранную из статьи.

```
EntryInformation ::= SEQUENCE {
        name                        Name,
        fromEntry                   BOOLEAN DEFAULT TRUE,
        information                 SET SIZE (1 ..MAX) OF CHOICE {
                attributeType           AttributeType,
                attribute               Attribute } OPTIONAL,
        incompleteEntry     [3]     BOOLEAN DEFAULT FALSE, -- нет в системах первого выпуска
        partialName         [4]     BOOLEAN DEFAULT FALSE, -- нет в системах первого или второго
выпусков
        derivedEntry        [5]     BOOLEAN DEFAULT FALSE -- нет в системах четвертого выпуска -- }
```

Параметр **Name** указывает Выделенное имя статьи или имя псевдонима для статьи. Выделенное имя статьи выдается всякий раз, когда это разрешено стратегией управления доступом. Если доступ разрешен к атрибутам статьи, но не к ее Выделенному имени, то Справочник может выдать либо уведомление об ошибке, либо имя действительного псевдонима для статьи.

Первичное Выделенное имя используется для параметра **Name**. Это означает, что если RDN, формируя имя, содержит атрибут, который имеет несколько выделенных значений, отличающихся контекстом, то первичное выделенное значение используется в качестве **value** в **AttributeTypeAndDistinguishedValue** выдаваемого RDN для этого атрибута. Так как для каждого RDN выданное **value** всегда является первичным выделенным значением, то компонент **primaryDistinguished** должен быть опущен для всех **AttributeTypeAndDistinguishedValue**.

Имена RDN в **Name** должны содержать альтернативные выделенные значения только в случаях, когда контекстный выбор применялся к выдаваемой информации статьи. Альтернативные выделенные значения выдаются как часть **valuesWithContext** в **AttributeTypeAndDistinguishedValue** выдаваемого RDN. Контекстные выборы, примененные к выдаваемой информации статьи (см. 7.6.1), применяются также к альтернативным выделенным значениям для определения выделенных значений, которые должны использоваться в **valuesWithContext**.

ПРИМЕЧАНИЕ 1. – Контекстный выбор не применяется к первичным выделенным значениям, выдаваемым в **Name**.

Если запрос был сделан для того, чтобы получить контекстную информацию вместе с результатом, то контекстная информация должна также выдаваться, если она имеется для выделенного значения в **Name** (используя элемент **valuesWithContext** имен RDN). Если выдаются альтернативные выделенные значения, то контекстная информация всегда выдается для всех выделенных значений.

ПРИМЕЧАНИЕ 2. – Если статья была размещена с использованием псевдонима, то такой псевдоним считается действительным псевдонимом. Для других случаев вопрос об действительности псевдонима находится вне предмета рассмотрения этих спецификаций Справочника.

ПРИМЕЧАНИЕ 3. – Для случаев, когда конкретный компонент Справочника имеет возможность выбора имен псевдонимов для выдачи, рекомендуется, чтобы он выбирал, если это возможно, одно и то же имя псевдонима для повторных запросов от того же инициатора, чтобы обеспечивать непротиворечивость службы.

Параметр **fromEntry** указывает, была ли информация получена из статьи (**TRUE**) или из копии статьи (**FALSE**).

Параметр **information** включается в тех случаях, когда выдается информация атрибута из статьи; он содержит набор соответствующих **attributeTypes** и **attributes**.

Параметр **incompleteEntry** включается и устанавливается в **TRUE** всякий раз, когда выдаваемая информация статьи является неполной в сравнении с запросом пользователя, например потому, что атрибуты или значения атрибутов опущены по причинам управления доступом (а их наличие разрешено раскрывать), или присутствует неполная скопированная информация вместе с **copyShallDo**, или лимит на размер параметра **attributeSizeLimit** был превышен. Он не устанавливается в **TRUE**, если выдается имя псевдонима вместо Выделенного имени.

Справочник должен завершить фазу распознавания имени в операциях полностью (включая проверку всех соответствующих известных ссылок на знание, следующих за отсылками, и т. д.), прежде чем рассматривать параметр управления службы **partialNameResolution**. Если все параметры распознавания имени были выбраны и, по крайней мере, одно значение RDN было распознано, то параметр **partialName** включается и устанавливается в **TRUE**, если запрос имел набор параметров управления службой **partialNameResolution**, а Справочник оказался неспособен завершить распознавание имени во всех RDN соответствующей статьи. Когда параметр **partialName** выдан со значением **TRUE**, он указывает, что информация взята из статьи в пункте, где последнее RDN было успешно разрешено.

Параметр **derivedEntry** включается и устанавливается в **TRUE** всякий раз, когда выдаваемая информация статьи содержит объединенные результаты, полученные объединением данных из более чем одной статьи справочника. Когда этот параметр установлен в **TRUE**, значение в **name** может быть именем любой соответствующей статьи, из которой получена информация, или именем псевдонима для любой из этих статей. Значение в **name** не должно использоваться в последующих операциях. Если параметр **derivedEntry** установлен в **TRUE**, а ответ подписан, то эта подпись будет того DSA, который выполнил объединение.

7.7.2 Информация семейства в информации статьи

Если должна выдаваться информация из составной статьи, то атрибуты каждого выдаваемого члена выбираются согласно **EntryInformationSelection** (который, возможно, изменен управляющим правилом поиска). Когда в запросе **search** установлена опция управления поиском **separateFamilyMembers**, каждый член выдается как отдельная статья. В других случаях, если должно выдаваться более одного члена, то информация статьи должна быть упакована таким способом, чтобы информация как бы исходила из одной статьи, которая может быть порождающим или подчиненным членом (последнее возможно, когда базовый объект запроса **search** – это член семейства, подчиненный порождающему члену, а этот порождающий член не был выбран параметром **FamilyReturn**). Атрибуты других членов должны быть упакованы в полученный атрибут **family-information** описанным ниже способом.

ПРИМЕЧАНИЕ 1. – Согласно этому несколько членов семейства всегда упаковываются в результат операций Чтение и Модификация статьи.

Полученный атрибут **family-information** используется только для упаковки; этот атрибут не существует как отдельный объект; он не может быть непосредственно выбран при помощи **entryInformationSelection** (любая попытка сделать это должна игнорироваться) и не может быть прямо защищен посредством управления доступом.

```
family-information ATTRIBUTE  ::=  {
        WITH SYNTAX                 FamilyEntries
        USAGE                       directoryOperation
        ID                          id-at-family-information }

FamilyEntries  ::= SEQUENCE  {
        family-class                OBJECT-CLASS.&id, -- значение структурного объектного класса
        familyEntries               SEQUENCE OF FamilyEntry }

FamilyEntry  ::= SEQUENCE  {
        rdn                     RelativeOistinguishedName,
        information             SEQUENCE OF CHOICE {
                attributeType           AttributeType,
                attribute               Attribute},
        family-info             SEQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL }
```

Атрибут **family-information** – это атрибут, содержащий несколько значений. Если порождающий член обозначен как источник информации, то каждое значение атрибута содержит информацию из одного семейства. Если член семейства,

подчиненный порождающему члену, обозначен как источник информации, то информация сортируется по значениям атрибутов на основе структурных объектных классов непосредственно подчиненных членов обозначенного члена. Каждый выбранный член семейства представлен значением типа **FamilyEntry**, который содержит:

- информацию выбранного атрибута (если она подходит) в виде типа атрибута или всего атрибута, в зависимости от значения **infoTypes** в **EntryInformationSelection**;

  ПРИМЕЧАНИЕ 2. – Как показано в п. 7.6, указание **infoTypes** применяется только к атрибутам-контейнерам, но не к самому атрибуту **family-information**.

- любую вложенную информацию **FamilyEntries** в виде полного атрибута **family-information**, собранного согласно структурным объектным классам подчиненных статей;

- невыбранные статьи не представляются, если они не являются порождающими членами для одного или более членов семейства, которые были выбраны.

7.8 Фильтр

7.8.1 Параметр Фильтр

Параметр **Filter** образует тест, которому удовлетворяет или не удовлетворяет конкретная статья. Фильтр выражается в форме проверок присутствия или значения определенных атрибутов статьи, причем тест завершается успешно, если и только если он оценивается значением TRUE.

ПРИМЕЧАНИЕ. – Фильтр может принимать значения TRUE (ИСТИНА), FALSE (ЛОЖЬ) или UNDEFINED (НЕОПРЕДЕЛЕННО).

```
Filter ::= CHOICE {
        item    [0]     FilterItem,
        and     [1]     SET OF Filter,
        or      [2]     SET OF Filter,
        not     [3]     Filter}
FilterItem ::= CHOICE {
        equality                [0]     AttributeValueAssertion,
        substrings              [1]     SEQUENCE {
                type                            ATTRIBUTE.&id ({ SupportedAttributes }),
                strings                         SEQUENCE OF CHOICE {
                        initial                 [0]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        any                     [1]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        final                   [2]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        control                         Attribute } }, -- используется для спецификации определения
                                                                       следующих параметров
        greaterOrEqual          [2]     AttributeValueAssertion,
        lessOrEqual             [3]     AttributeValueAssertion,
        present                 [4]     AttributeType,
        approximateMatch        [5]     AttributeValueAssertion,
        extensibleMatch         [6]     MatchingRuleAssertion,
        contextPresent          [7]     AttributeTypeAssertion}

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule            [1]     SET SIZE (1..MAX) OF MATCHING-RULE.&id,
        type                    [2]     AttributeType OPTIONAL,
        matchValue              [3]     MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                        -- matchValue должно быть значением типа, определенного полем &AssertionType
                        -- одного из информационных объектов MATCHING-RULE, определяемого правилом сопоставления -- } ),
        dnAttributes            [4]     BOOLEAN DEFAULT FALSE}
```

Filter представляет собой либо **FilterItem** (см. п. 7.8.2), либо выражение, охватывающее более простые фильтры, соединенные логическими операторами **and** (и), **or** (или)и **not** (не). На работу фильтра может воздействовать стратегия ослабления, которая может вызывать замену одного правила сопоставления на другое или может устанавливать значения, которые должны рассматриваться для сопоставления.

Filter, представляющий собой **FilterItem**, имеет значение этого **FilterItem** (т. е. TRUE, FALSE или UNDEFINED).

Filter, являющийся оператором **and** для набора фильтров, имеет значение TRUE, если этот набор – пустой или если каждый фильтр имеет значение TRUE; он имеет значение FALSE, если, по крайней мере, один фильтр имеет значение FALSE; в остальных случаях (т. е. когда, по крайней мере, один фильтр имеет значение UNDEFINED, но ни один из фильтров не имеет значение FALSE) его значением будет UNDEFINED.

Filter, являющийся оператором **or** для набора фильтров, имеет значение FALSE, если этот набор – пустой или если каждый фильтр имеет значение FALSE; он имеет значение TRUE, если, по крайней мере, один фильтр имеет значение TRUE; в остальных случаях (т. е. когда, по крайней мере, один фильтр имеет значение UNDEFINED, но ни один из фильтров не имеет значение TRUE) его значением будет UNDEFINED.

Filter, являющийся оператором **not** для некоторого фильтра, будет иметь значение TRUE, если этот фильтр имеет значение FALSE; будет иметь FALSE, если он имеет значение TRUE; и будет иметь UNDEFINED, если он имеет значение UNDEFINED.

Неинвертированный элемент фильтра определяется как элемент, вложенный в пределах четного числа (возможно нуль) элементов **not** в рамках самого внешнего **Filter**. Таким образом, фильтр, содержащий только элементы фильтра в комбинации с **and** или **or**, имеет только *неинвертированные* элементы. *Инвертированный* элемент фильтра определяется как элемент, вложенный в пределах нечетного числа элементов **not** в рамках самого внешнего **Filter**.

7.8.2 Элемент фильтра

FilterItem – это утверждение (результат проверки) о наличии атрибута(ов) или о значении(ях) атрибута в тестируемой статье. Утверждение о конкретном типе атрибута удовлетворяется также в том случае, когда статья содержит подтип атрибута и утверждение равно TRUE для этого подтипа, а опция управления службы **noSubtypeMatch** не установлена, либо когда имеется совокупный атрибут статьи (см. п. 7.6), для которого утверждение равно TRUE или если:

- опция управления услугой **dontMatchFriends** не установлена; и
- статья содержит атрибут "друга" для конкретного атрибута, который имеет правило согласования, совместимое с утверждением; и
- утверждение равно TRUE для атрибута "друга".

Каждое утверждение может быть TRUE, FALSE или UNDEFINED.

Каждый **FilterItem** содержит или подразумевает один или несколько **AttributeTypes**, которые определяют конкретный рассматриваемый атрибут(ы).

Любое утверждение о значениях такого атрибута определено только тогда, когда **AttributeType** известен механизму оценки, предполагаемый(е) **AttributeValue** соответствует(ют) синтаксису атрибута, определенному для этого типа атрибута, применяемое или указанное правило сопоставления применимо к этому типу атрибута, а предоставленный **matchValue** (когда используется) соответствует синтаксису, определенному для указанных правил сопоставления. Когда эти условия не выполняются, **FilterItem** установит оценку в логическое значение UNDEFINED.

ПРИМЕЧАНИЕ 1. – Ограничения на управление доступом могут влиять на оценку **FilterItem** и могут заставить **FilterItem** выдать UNDEFINED.

Утверждение, которое определено этими условиями, может дополнительно дать оценку UNDEFINED если оно касается значения атрибута, а тип атрибута не присутствует в атрибуте, для которого утверждение тестируется. Утверждение, которое определено этими условиями и касается присутствия типа атрибута, оценивается как FALSE.

Утверждения о значении атрибута в элементах фильтра оцениваются с использованием правил сопоставления, определенных для этого типа атрибута и заменяемых, когда необходимо, в соответствии с действием стратегии ослабления. Утверждения в правилах сопоставления оцениваются так, как указано в их определении. Правило сопоставления, определенное для конкретного синтаксиса, может использоваться только для выполнения утверждений о атрибутах этого синтаксиса или подтипах этого синтаксиса.

ПРИМЕЧАНИЕ 2. – Действие стратегии ослабления может заставить конкретное правило сопоставления возвратиться к правилу сопоставления **nullMatch,** которое всегда оценивает как TRUE (если неинвертированный элемент) или FALSE (если инвертированный) – см. п. 6.7.2 Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6.

FilterItem может быть равен UNDEFINED (как описано выше). В других случаях, когда **FilterItem** проверяет:

a) **equality** – результатом будет TRUE, если и только если существует значение атрибута или значение одного из его подтипов, для которых правило сопоставления **equality** применимо к этому значению, а представленное значение дает TRUE.

b) **substrings** – результатом будет TRUE, если и только если существует значение атрибута или одного из его подтипов, для которых правило сопоставления **substring** применено к этому значению, а представленное значение в **strings** дает TRUE. См. Рекомендацию МСЭ-Т X.520 | ИСО/МЭК 9594-6 с описанием семантики представленного значения.

c) **greaterOrEqual** – результатом будет TRUE, если и только если существует значение атрибута или одного из его подтипов, для которых правило сопоставления **ordering** применимо к этому значению, а представленное значение дает FALSE, т. е. имеется значение атрибута, которое *больше* представленного значения *или равно* ему.

d) **lessOrEqual** – результатом будет TRUE, если и только если существует значение атрибута или одного из его подтипов, для которых правило сопоставления **equality** *или* правило сопоставления **ordering** применимо к этому значению, а представленное значение дает TRUE, т. е. имеется значение атрибута, которое *меньше* представленного значения *или равно* ему.

e) **present** – результатом будет TRUE, если и только если атрибут или один из его подтипов присутствует в статье.

f) **approximateMatch** – результатом будет TRUE, если и только если имеется значение атрибута или одного из его подтипов, для которых некоторый местный алгоритм приблизительного сопоставления (например, с вариантами орфографии, фонетическим соответствием и т. д.) дает TRUE. Если элемент оценивается на предмет эквивалентности, то он должен также удовлетворять приблизительному соответствию. В этом издании спецификации Справочника не дается других рекомендаций по приблизительному сопоставлению". Если приблизительное сопоставление не поддерживается, то этот **FilterItem** должен быть обработан как соответствие **equality**.

g) **extensibleMatch** – результатом будет TRUE, если и только если существует значение атрибута с указанным **type** или значение одного из его подтипов, для которых правило сопоставления, указанное в **matchingRule**, применено к этому значению, а представленное значение в **matchValue** дает TRUE.
Если даны несколько правил сопоставления, то способ их объединения в новое правило не определяется (этот способ является местным алгоритмом, который отражает семантику составных правил сопоставления, например, соответствие **phonetic + keyword**).

Если компонент **type** опущен, то сравниваются все типы атрибута, которые совместимы с этим правилом сопоставления. Если компонент **dnAttributes** равен **TRUE**, то атрибуты Выделенного имени статьи используются дополнительно к тем атрибутам статьи, для которых проводилась оценка соответствия.

Если в фильтре запрашивается **extensibleMatch** (что предпочтительнее, чем **extendedFilter**), то в параметре **criticalExtensions** информации **CommonArguments** должен быть установлен бит **extendedFilter**, указывающий, что расширение является критическим.

Если реализация не поддерживает никакое из правил сопоставления, определенных в подкомпоненте **matchingRule**, или если ни одно из правил сопоставления не совместимо с типом атрибута, то элемент фильтра **extensibleMatch** дает оценку UNDEFINED, когда опция управления поиском **performExactly** не установлена. Когда опция управления поиском **performExactly** установлена, запрос **search** отклоняется с:

– **serviceError**, указывающим проблему **unsupportedMatchingUse**;

– атрибутом уведомления **searchServiceProblem** со значением **id-pr-unsupportedMatchingRule**, если все правила сопоставления не поддерживаются, а в остальных случаях со значением **id-pr-unsupportedMatchingUse**;

– атрибутом уведомления **attributeTypeList**, имеющим в качестве значения тип атрибута, для которого были определены недействительные правила сопоставления; и

– атрибутом уведомления **matchingRuleList**, имеющим в качестве значений идентификаторы объектов для неподдерживаемых и/или несовместимых правил сопоставления.

ПРИМЕЧАНИЕ 3. – Компонент **extensibleMatch** не допускается для систем первого издания.

h) **contextPresent** – результатом будет TRUE, если и только если **AttributeTypeAssertion** для этого типа атрибута либо для одного из его подтипов, когда опция управления службой **noSubtypeMatch** не установлена, дает TRUE.

Если контекстные утверждения включены в утверждение значения атрибута в элементе фильтра, то этот элемент фильтра оценивается только относительно тех значений, которые удовлетворяют всем заданным контекстным утверждениям, описанным в п. 8.9.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если контекстные утверждения не включены в утверждение значения атрибута, то контекстные утверждения "по умолчанию" должны применяться так, как описано в п. 8.9.2.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

7.8.3 Оценивающие фильтры с информацией семейства

Конкретные группировки семейств при выполнении требований фильтра работают следующим образом:

Компонент **entryOnly** означает, что только те члены семейства, которые полностью выполняют требования фильтра, маркируются как содействующие и участвующие члены (см. определения содействующих и участвующих членов в п. 7.13).

Компонент **compoundEntry** означает, что вся составная статья формирует группу, которая должна удовлетворять всему фильтру; в пределах каждой составной статьи, которая удовлетворяет фильтру, члены семейства, содействующие соответствию, маркируются как содействующие члены, в то время как все члены составной статьи маркируются как участвующие статьи.

Компонент **strands** означает, что фильтр применяется к каждой полной трассе от листа до порождающего члена. Составная статья соответствует фильтру, если, по крайней мере, одна трасса соответствует фильтру. Члены семейства на соответствующей трассе, которые содействуют соответствию, маркируются как содействующие члены, в то время как все члены на соответствующей трассе маркируются как участвующие члены.

Трасса – это набор членов из семейства, которые формируют путь от члена-листа до порождающего члена, поэтому количество трасс соответствует количеству статей-листьев.

Компонент **multiStrand** означает, что комбинация, содержащая по одной трассе от каждого класса семейства, является семейством, сгруппированным для целей сопоставления. Все комбинации должны рассматриваться по одной за один раз. Составная статья соответствует фильтру, если, по крайней мере, одна комбинация трасс соответствует фильтру. Члены семейства на соответствующей комбинации трасс, которые содействуют соответствию, маркируются как содействующие члены, в то время как все члены соответствующей комбинации трасс маркируются как участвующие члены.

Две трассы имеют одинаковый *класс семейства*, если и только если члены семейства, которые являются непосредственно подчиненными по отношению к порождающему элементу, имеют одинаковый структурный класс объектов.
Трасса *соответствует* фильтру, если и только если она присутствует, по крайней мере, в одной из всех возможных комбинаций трасс, которая определяет, что статья соответствует подфильтру. Отсюда следует:

– Если порождающий элемент соответствует подфильтру полностью, то соответствуют *все* трассы.

– Аналогично, если имеются три класса семейства для конкретного порождающего элемента и два класса из них соответствуют подфильтру без рассмотрения третьего, то все трассы третьего класса семейства соответствуют подфильтру.

Компонент **multiStrand** применим только тогда, когда базовый объект – это порождающий элемент (или выше) в DIT. Если базовый объект – это член семейства, но не порождающий элемент, то **multiStrand** игнорируется, а **entryOnly** подставляется.

7.9 Результаты поиска в виде страниц

DUA использует параметр **PagedResultsRequest** для запроса результатов операции Список или Поиск в виде "страницы за страницей": DUA запрашивает у DSA выдавать только поднабор – *страницу* – результатов операции, в частности, следующие подчиненные **pageSize** или статьи, и выдавать **queryReference**, которая может использоваться, чтобы запросить следующий набор результатов в последующем запросе.

Результаты в виде страниц могут производиться агентом DSA, к которому DUA был привязан с помощью операции привязки (связанный DUA), или агентом DSA, начавшим этап первоначальной оценки (первоначальный исполнитель, подробно описанный в п. 15.5.5 Рек. МСЭ-Т X.518 | ИСО/МЭК 9594-4).

Этот параметр не должен использоваться, если результаты должны быть подписаны; за исключением случая существования понимания между агентами DSA, взаимодействующих с целью предоставления результатов в виде страниц, о том, что предоставляющий результаты DSA может изъять подписи на результатах, полученных от других DSA, и затем самостоятельно подписать результаты, выдаваемые для DUA. Способ установления понимания находится вне сферы применения настоящей спецификации Справочника. Хотя DUA может запрашивать **pagedResults**, в DSA разрешается игнорировать такой запрос и выдавать ответ на него обычным способом.

ПРИМЕЧАНИЕ 1. – Результат может быть непредсказуемым в случае конфигурации, не являющейся "как следует подключенной", например, когда из-за затенения и использования NSSR распознавание имени будет локализовывать более одного базового объекта.

Если запрашиваются результаты в виде страниц и осуществляется вызов страниц, то DSA, вызывающий страницы, игнорирует управление услугой **sizeLimit** при ее наличии. Если вызов страниц не осуществляется, управление услугой **sizeLimit** должно выполняться. Содействующий DSA (см. п. 15.5.5 Рек. МСЭ-Т | ИСО/МЭК 9594-4) выполняет управление услугой **sizeLimit**.

```
PagedResultsRequest ::= CHOICE {
        newRequest                  SEQUENCE {
                pageSize                    NTEGER,
                sortKeys                    SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
                reverse             [1]     BOOLEAN DEFAULT FALSE,
                unmerged            [2]     BOOLEAN DEFAULT FALSE,
                pageNumber          [3]     INTEGER OPTIONAL },
        queryReference              OCTET STRING,
        abandonQuery        [0]     OCTET STRING }

SortKey ::= SEQUENCE {
        type                AttributeType,
        orderingRule        MATCHING-RULE.&id OPTIONAL }
```

Для новой операции Список или Поиск **PagedResultsRequest** устанавливается в **newRequest**, который состоит из следующих параметров:

a) Параметр **pageSize** указывает максимальное число подчиненных элементов или статей, которые будут выдаваться в результатах. DSA должен выдавать не больше этого запрошенного числа статей. Параметр **sizeLimit**, если он установлен, игнорируется. Включаемая информация семейства не подгоняется к размеру страницы, когда она упаковывается в полученные атрибуты **family-information**.

b) Параметр **sortKeys** определяет последовательность типов атрибута и необязательные правила сопоставления порядка следования для использования в качестве ключей сортировки при сортировке выдаваемых статей до их выдачи к DUA. В случае операций Списка сортировка осуществляется RDN, однако требования к сортировке применяются только к атрибутам, содержащимся в RDN. В случае операций Поиска порядок следования применяется только к атрибутам, которые реально предоставляются (как результат выбора и управления доступом с сортировкой по выделенному имени в качестве возможности восстановления). Статьи сортируются согласно их значениям и атрибуту **type** первого **SortKey** в последовательности, а в случае нескольких статей, имеющих одинаковую позицию сортировки, следующего **SortKey** в последовательности и так далее.

   Для конкретного **SortKey** DSA использует правило сопоставления **orderingRule**, если оно имеется, а в остальных случаях использует правило сопоставления **ordering** атрибута, если он определен; DSA игнорирует ключ сортировки, если ни один из ключей не определен. Если тип атрибута имеет несколько значений, то используется "наименьшее" значение; если тип атрибута отсутствует в выдаваемых результатах, то он рассматривается как "наибольшее" для всех других сопоставляемых значений. Разрешается поддерживать в DSA только некоторые последовательности ключей сортировки (таким образом, DSA, который содержит и выдает свои данные согласно внутреннему порядку " по алфавиту фамилий", будет способен работать только с одной последовательностью ключей сортировки). Если он не может поддерживать запрошенную последовательность, то он должен использовать последовательность сортировки "по умолчанию".

   Иерархическая группа не должна быть отделена, а выдается в последовательность, как описано в п. 10.3 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2. При выполнении сортировки первая статья иерархической группы, которая должна быть выдана, определяет положение иерархической группы в выводимом результате.

   ПРИМЕЧАНИЕ 2. – Иерархическая группа может соединять страницы.

c) Если параметр **reverse** равен **TRUE**, то DSA будет выдавать отсортированные результаты в обратном порядке (т. е. от "наибольшего" к "наименьшему" – если тип атрибута имеет несколько значений, то используется "наибольшее" значение; если тип атрибута отсутствует в выдаваемых результатах, то он рассматривается как "наименьшее" для всех других сопоставляемых значений). Если этот параметр равен **FALSE**, то DSA выдает результаты в прямом порядке. Если параметр **sortKeys** не определен, то этот параметр игнорируется.

d) Если параметр **unmerged** равен **TRUE**, а DSA, ответственный за вызов страниц, собирает результаты ряда других DSA, то он должен выдавать все данные от одного DSA (в отсортированном порядке), а затем выдавать данные от следующего DSA. Если этот параметр равен **FALSE**, то DSA должен объединить результаты от всех других DSA и сортировать объединенные данные перед их выдачей. Если параметр **sortKeys** не определен, то этот параметр игнорируется. Семантика параметра **unmerged** не изменяется в зависимости от того, поддерживает ли DSA результаты в виде страниц DSP или нет.

e) Если имеется параметр **pageNumber**, то он указывает на то, что пользователь желает начать с конкретной страницы, а не обязательно с первой страницы. Параметр игнорируется, если порядок следования не определяется.

Для следующего запроса, т. е. для запроса следующего набора результатов в виде страниц, DUA делает такой же запрос на Список или Поиск, как и ранее, но устанавливает параметр **PagedResultsRequest** в вариант **queryReference** с тем значением этого параметра, какой был выдан в **PartialOutcomeQualifier** предыдущих результатов. DUA не имеет сведений о **queryReference**, который является доступным в DSA для использования, когда он желает сделать запись контекстной информации для запроса. DSA использует указанную информацию для определения того, какие результаты следует выдавать далее.

В любое время DUA может указать путем установки набора **PagedResultsRequest** на **abandonQuery** со значением, идентичным значению **queryReference**, выдаваемомому в **PartialOutcomeQualifier** предыдущих результатов, что при осуществлении запроса того же, что и прежде, **Списка** или **Поиска** не требуется больше страниц. Дополнительные страницы не должны запрашиваться и выдаваться. Реализация определяет, когда страницы будут стираться.

В случае, если выбран **queryReference** или **abandonQuery**, новый запрос и исходная информация должны быть идентичны в следующих отношениях:

- **baseObject** внутри **SearchArgument** или **object** внутри **ListArgument** соответствуют настоящему и исходному запросам;
- субкомпонент **queryReference** в **pagedResults** должен быть идентичен значению **queryReference**, выдаваемому в **PartialOutcomeQualifier** предыдущего результата;
- компонент опций типа данных **ServiceControls** устанавливает идентичные опции для настоящего и исходного запросов;
- **operationProgress** (если имеется) должен быть идентичен для настоящего и исходного запросов.

В ином случае выдается **serviceError** с сообщением о проблеме **invalidQueryReference**.

ПРИМЕЧАНИЕ 3. – Если в DIB изменяется порядок следования запросов на поиск, то DUA может не видеть влияния этих изменений. Это зависит от реализации.

ПРИМЕЧАНИЕ 4. – Ссылка на запрос может оставаться действительной даже тогда, когда DUA начинает новую операцию Список или Поиск. DUA может запрашивать результаты в виде страниц с помощью нескольких запросов, а затем возвращаться к более раннему запросу и запрашивать следующую страницу результатов, используя ссылку на запрос, которая была для этого использована. Число "активных" ссылок на запрос, к которым DUA может возвращаться, является местной опцией реализации DSA, как и "срок жизни" этих ссылок на запрос.

ПРИМЕЧАНИЕ 5. – Поддержка выбора **abandonQuery** имеется только для систем после четвертого выпуска.

ПРИМЕЧАНИЕ 6. – Когда прекращает действие ассоциация DAP, теряется доступ ко всем связанным с ней результатам в виде страниц. Доступ к результатам в виде страниц может быть осуществлен только в пределах ассоциации DAP, в рамках которой они изначально были вызваны.

7.10 Параметры безопасности

SecurityParameters управляют выполнением различных мер безопасности, связанных с работой Справочника.

ПРИМЕЧАНИЕ 1. – Эти параметры передаются от отправителя к получателю. Если такие параметры появляются в аргументе операции, то инициатор – это отправитель, а исполнитель – получатель. При передаче результата роли меняются.

```
SecurityParameters ::= SET {
    certification-path          [0]  CertificationPath          OPTIONAL,
    name                        [1]  DistinguishedName          OPTIONAL,
    time                        [2]  Time                       OPTIONAL,
    random                      [3]  BIT STRING                 OPTIONAL,
    target                      [4]  ProtectionRequest          OPTIONAL,
    respons                     [5]  BIT STRING                 OPTIONAL,
    operationCode               [6]  Code                       OPTIONAL,
    attribiiteCertificationPath [7]  AttributeCertificationPath OPTIONAL,
    errorProtection             [8]  ErrorProtectionRequest     OPTIONAL,
    errorCode                   [9]  Code                       OPTIONAL}

ProtectionRequest ::= INTEGER { none (0), signed (1)}

Time ::= CHOICE {
    utcTime          UTCTime,
    generalizedTime  GeneralizedTime}

ErrorProtectionRequest ::= INTEGER { none (0), signed (1)}
```

Компонент **CertificationPath** – это последовательность, содержащая сертификат подписывающего пользователя, и необязательная последовательность из одного или более сертификатов органа сертификации (CA). (См. раздел 7 в Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8.) Сертификат пользователя применяется для связывания открытого ключа и выделенного имени подписывающего лица и может применяться для проверки подписи в аргументе запроса, ответе или уведомлении об ошибке. Этот параметр должен присутствовать и содержать сертификат подписывающего пользователя, если аргумент запроса, ответ или уведомление об ошибке подписаны. Дополнительные сертификаты могут присутствовать и могут использоваться для определения действительности сертификата подписывающего пользователя. Дополнительные сертификаты не требуются, если получатель связан с тем же органом сертификации, что и подписывающее лицо. Если получатель требует траекторию сертификации для подтверждения, а соответствующий параметр отсутствует, то получатель либо отклоняет подпись, либо пытается определить траекторию сертификации; выбор определяется местным решением.

Компонент **name** – это выделенное имя первого предполагаемого получателя аргумента или результата. Например, если DUA создает подписанный аргумент, то имя является выделенным именем того DSA, которому операция предназначена.

ПРИМЕЧАНИЕ 2. – Если первый предполагаемый получатель имеет альтернативные выделенные имена, различающиеся контекстом, то **name** может быть одним из альтернативных имен. Однако идентификация и управление доступом, которые могут быть основаны на значении **name**, не смогут работать нужным образом, если не используется первичное выделенное имя.

Компонент **time** – это предполагаемый предельный интервал времени действительности запроса, ответа или уведомления об ошибке. Он используется совместно со случайным числом, чтобы иметь возможность обнаруживать атаки подмены.

Значение компонента **random** – это число, которое должно отличаться для каждого запроса, ответа или уведомления об ошибке. Оно используется вместе с параметром времени, чтобы иметь возможность обнаруживать атаки подмены. Если требуется целостность последовательности, то может использоваться аргумент со случайным значением, чтобы переносить число целостности последовательности следующим образом:

a) Случайное значение, используемое с аргументами операции, образуется применением заранее согласованной последовательности (например, предыдущее значение плюс 1) из следующего источника:

   i) для первой операции, посланной из системы для привязывания, случайное значение передается из удаленной равноправной системы в аргументе/результате операции Привязывание; и;

   ii) для последующих операций случайное значение передается при предыдущей операции в том же направлении.

b) Случайное значение, используемое с результатами операции или уведомлениями об ошибке, образуется применением заранее согласованной последовательности из случайного значения в запросе (например, случайное значение в аргументе запроса плюс 1).

Компонент **target** в виде **ProtectionRequest** может появиться только в запросе на подлежащую выполнению операцию; он характеризует предпочтение инициатора относительно степени защиты, которая должна быть обеспечена для результата. Обеспечиваются два уровня: **none** (нет запроса на защиту – значение "по умолчанию") и **signed** (от Справочника требуется подписывать результат). Степень защиты, фактически обеспеченная для результата, определяется формой результата и может быть равна или ниже запрошенной, что зависит от ограничений Справочника.

Компонент **response** используется для передачи любой информации обратно, к инициатору запроса.

Компонент **operationCode** используется для безопасного привязывания кода операции к аргументам запроса, результатам или уведомлениям об ошибке.

Компонент **attributeCertificationPath** используется для переноса свидетельства безопасности для управления доступом на основе правила или переноса другого атрибута в Сертификате атрибута, возможно, вместе с сертификатами, необходимыми для проверки действительности Сертификата атрибута.

Компонент **errorProtection** может появиться только в запросе на подлежащую выполнению операцию; он характеризует предпочтение инициатора относительно степени защиты, которая должна быть обеспечена для уведомления об ошибке. Обеспечиваются два уровня: **none** (нет запроса на защиту – значение "по умолчанию") и **signed** (от Справочника требуется подписывать уведомление об ошибке). Степень защиты, фактически обеспеченная для уведомления об ошибке, определяется формой уведомления и может быть равна или ниже запрошенной, что зависит от ограничений Справочника.

ПРИМЕЧАНИЕ 3. – DUA может требовать, чтобы контекст метки защиты выдавался со значением атрибута, использующим контекстный выбор.

Компонент **errorCode** используется для защиты кода ошибки при выдаче уведомления об ошибке в ответе операции.

Если синтаксис параметра **Time** был выбран в виде типа **UTCTime**, то значение поля года с двумя цифрами должно переводиться в четырехразрядное значение года следующим образом:

- Если 2-разрядное значение равно от 00 до 49 включительно, то к значению нужно добавить 2000.
- Если 2-разрядное значение равно от 50 до 99 включительно, то к значению нужно добавить 1900.

Компонент **GeneralizedTime** будет использоваться, если выбранная версия равна **v2** или больше. Использование **GeneralizedTime** при выбранной версии **v1** может препятствовать взаимодействию с реализациями, не имеющими возможности выбора между **UTCTime** и **GeneralizedTime**. Эта ответственность лежит на тех, кто определяет области, в которых эта спецификация Справочника будет использоваться, например, при выделении групп, в которых **GeneralizedTime** может использоваться с определенного времени. Ни в каком случае **UTCTime** не будет использоваться для представления дат свыше 2049 года.

7.11 Общие элементы процедуры для управления доступом

В данном подразделе определяются элементы процедуры, которые являются общими для всех операций абстрактной службы, когда действует **basic-access-control** (управление базовым доступом), или **rule-based-access-control** (управление доступом на основе правила), или оба вместе. Если действуют оба механизма, то порядок, в котором они применяются, является местным вопросом, за исключением следующего: если каким-либо механизмом отклонен доступ к статье, к типу атрибута или к значению атрибута, то согласие от другого механизма не должно отменять это решение. В частности, разрешение *DiscloseOnError (Раскрытие в уведомлении об ошибке)* механизма **basic-access-control** является согласием, которое не должно отменять запрет механизма **rule-based-access-control**.

7.11.1 Общие элементы процедуры для управления базовым доступом

7.11.1.1 Переименование псевдонима

Если в процессе размещения целевой статьи объекта (указанной в аргументе операции абстрактной службы) требуется переименование псевдонима, то никаких специальных разрешений не требуется для выполнения переименования псевдонима. Однако если переименование псевдонима может привести к выдаче **ContinuationReference** (т. е. отсылки **Referral**), то применяется следующая последовательность параметров управления доступом. Если DSA привязывает (сцепляет) запрос к другому DSA и получает от него в ответ отсылку, то параметры управления доступом должны применяться к этой отсылке, если параметр **targetObject** в отсылке является тем же, что и в привязанном запросе. Таким образом, DSA будет отслеживать все отсылки независимо о того, были они созданы на месте или удаленно.

1) Разрешение *Чтение* требуется для статьи псевдонима. Если разрешение не предоставляется, то операция получает отказ согласно процедуре, описанной в п. 7.11.1.

2) Разрешение *Чтение* требуется для атрибута **aliasedEntryName** и для единственного значения, которое содержится в нем. Если разрешение не предоставляется, то операция получает отказ и выдается параметр **nameError** с указанием причины **aliasDereferencingProblem**. Элемент **matched** должен содержать имя статьи псевдонима.

ПРИМЕЧАНИЕ. – В дополнение к параметрам управления доступом, описанным выше, стратегия безопасности может препятствовать раскрытию информации, которая иначе была бы передана как **ContinuationReference** в **Referral**. Если такая стратегия действует и если DUA ограничивает службу, указав **chainingProhibited**, то Справочник может выдать параметр **serviceError** с указанием причины **chainingRequired**. В остальных случаях должен выдаваться параметр **securityError** с указанием причины **insufficientAccessRights** или **noInformation**.

7.11.1.2 Выдача уведомления об ошибке имени

Если при выполнении операции абстрактной службы указанный целевой объект (псевдоним или статья) – например, Имя статьи, которую нужно прочитать, или **baseObject** в запросе **search** – не может быть найден, то должен быть выдан параметр **nameError** с указанием проблемы **noSuchObject**. Элемент **matched** должен содержать либо имя следующей старшей статьи, для которой разрешение *DiscloseOnError* предоставляется, либо имя корня DIT (т. е. пустую последовательность **RDNsequence**).

ПРИМЕЧАНИЕ. – Второй вариант может использоваться в DSA, который не имеет доступа ко всем старшим статьям.

7.11.1.3 Отказ в раскрытии наличия статьи

Если доступ отклонен процедурой управления **rule-based-access-control**, то разрешение *DiscloseOnError* не применимо.

Если при выполнении операции абстрактной службы необходимое разрешение на уровне статьи не предоставляется для указанной статьи целевого объекта – например, для статьи, которую требуется читать, – то операция получает отказ и выдается одно из следующих уведомлений об ошибке: если разрешение *DiscloseOnError* предоставляется для целевой статьи, то должен выдаваться параметр **securityError** с указанием проблемы **insufficientAccessRights** или **noInformation**. В остальных случаях должен выдаваться **nameError** с указанием **noSuchObject**. Элемент **matched** должен содержать либо имя следующей старшей статьи, для которой разрешение *DiscloseOnError* предоставляется, либо имя корня DIT (т. е. пустую последовательность **RDNsequence**).

ПРИМЕЧАНИЕ. – Второй вариант может использоваться в DSA, который не имеет доступа ко всем старшим статьям.

Кроме того, всякий раз, когда Справочник обнаруживает ошибку в операции (включая отсылку), он должен гарантировать, что выдача с ошибкой не будет компрометировать наличие названной целевой статьи и любой из ее старших статей. Например, перед выдачей параметра **serviceError** с указанием проблемы **timeLimitExceeded** или **updateError** с указанием проблемы **notAllowedOnNonLeaf** Справочник проверяет, что разрешение *DiscloseOnError* предоставлено для целевой статьи. Если это не так, то должна следовать процедура, описанная в предыдущем абзаце.

7.11.1.4 Выдача Выделенного имени

В операциях Сравнение, Список и Поиск для статьи **object** (или **baseObject**) требуется разрешение *ReturnDN* (*выдача DN*), если в результате переименования псевдонима должно быть выдано выделенное имя объекта в параметре **name** результата операции (см. п. 9.2.3). Если это разрешение не предоставлено, то Справочник должен выдать имя псевдонима вместо статьи, как это описано в п. 7.7, или должен опустить параметр **name** целиком.

В операции Чтение или Поиск требуется разрешение *ReturnDN* для статьи, чтобы выдать ее выделенное имя в параметре **EntryInformation**. Если это разрешение не предоставляется, то Справочник должен выдать имя псевдонима вместо статьи, как это описано в п. 7.7, или если имя псевдонима не доступно, то прервать операцию и выдать параметр **nameError** (в случае операции Чтение), либо опустить статью в ответе (в случае операции Поиск).

Если предоставленное пользователем имя псевдонима выдается в ответе, то признак **aliasDeferenced** в **CommonResults** не должен быть установлен в **TRUE**.

7.11.2 Общие элементы процедуры для управления доступом на основе правила

7.11.2.1 Доступ к статье (разрешение на уровне статьи)

При доступе к статье требуется разрешение, чтобы обратиться, по крайней мере, к одному значению атрибута статьи. Если разрешение на уровне статьи не предоставлено, то должен выдаваться параметр **nameError** с указанием проблемы **noSuchObject**.

7.11.2.2 Выдача имени статьи

При выдаче DN статьи требуется разрешение, чтобы обратиться ко всем значениям атрибута, по крайней мере, одного контекстного варианта RDN статьи (это называется разрешением *RDN*). Никакие разрешения не требуются от старших статей этой статьи. Если разрешение *RDN* не предоставляется, то DSA может выбирать: выдать DN действительного псевдонима статьи, для которой разрешение *RDN* было предоставлено, либо опустить компонент Имя в результате операции.

ПРИМЕЧАНИЕ. – Выбор соответствующего имени псевдонима описан подробнее в примечаниях к п. 7.7.

7.11.2.3 Переименование псевдонима

При переименовании псевдонима требуется разрешение для доступа к значению атрибута **aliasedEntryName**.

7.11.2.4 Выдача уведомления об ошибке имени (noSuchObject)

В компонент **matched** параметра **nameError** с указанием проблемы **noSuchObject** будет записано имя следующей старшей статьи, для которой инициатор имеет разрешение *RDN*. Если такая статья не доступна для DSA, который выдает уведомление об ошибке, то должно выдаваться имя корня DIT.

7.11.2.5 Доступ к атрибуту

При доступе к атрибуту необходимо разрешение для доступа, по крайней мере, к одному из значений этого атрибута.

7.11.2.6 Удаление информации

Чтобы удалить значение атрибута, необходимо разрешение для доступа к этому значению. При удалении статьи или атрибута операция должна выдать успешный ответ, если, по крайней мере, одно значение атрибута удалено, независимо от того, сколько значений было запрошено удалить.

7.11.2.7 Вызов правил поиска

Чтобы применить правило поиска к аргументам операции Поиск, для инициатора операции Поиск требуется разрешение на вызов определенного правила поиска. Пользователь не нуждается в других разрешениях для доступа к атрибуту правила поиска или к подстатье, которая его содержит.

7.11.3 Информация семейства

Информация семейства рассматривается как любая другая информация, за исключением случая, когда для ACI (Access Control Information, информация управления доступом) параметр **ProtectedItem** отмечен как **includeFamily**; если ACI применим к порождающему элементу или к члену семейства, то это заставляет младших членов семейства подчиняться тому же самому ACI. Параметр **IncludeFamily** имеет смысл только тогда, когда применяется к защищенному элементу **entry**.

7.12 Административное управление информационным деревом DSA

Информационное дерево DSA, поддерживаемое в DSA, может управляться посредством определенной Абстрактной службы Справочника. При административном управлении Информационным деревом DSA:

- все DSE (DSA-Specific-Entry, специфичная для DSA статья), включая DSE корня, в DSA видимы через DAP (Directory Access Protocol, протокол доступа к Справочнику);
- атрибуты, определенные как не подлежащие модификации пользователем, можно модифицировать (хотя DSA может давать ответ, содержащий параметр **serviceError** с указанием проблемы **unwillingToPerform**, если он не может выполнить требуемое изменение);
- знания – это просто другой атрибут, который может быть прочитан и изменен; и
- DSA никогда не связывает запросы в цепочку и не выдает отсылки или ссылки на продолжение.

Видимость DSE, а также поиск или изменение атрибутов операции может управляться с помощью управления доступом обычным способом.

Управление Информационным деревом DSA производит DUA с использованием следующих процедур:

1) DUA привязывается непосредственно к DSA, содержащим Информационное дерево DSA, которое будет управляться;
2) при каждой операции, которая используется для управления Информационным деревом DSA:
   – устанавливается бит расширения **manageDSAIT**;
   – устанавливается опция **manageDSAIT**;
   – опция **manageDSAITPlaneRef** включается, если должен управляться конкретный уровень репликации;

   следующие компоненты игнорируются Справочником:
   – **operationProgress** в **CommonArgument**;
   – **referenceType** в **CommonArgument**;
   – **entryOnly** в **CommonArgument**;
   – **nameResolveOnMaster** в **CommonArgument**; и
   – **chainingProhibited** в **ServiceControls**.

7.13 Процедуры для семейств статей

Как определено в п. 7.3.2, члены семейства в составной статье могут быть сгруппированы вместе с целью проведения операции. Эта группировка действует только для операций Сравнение, Поиск и Удаление статьи. Если группировка семейства указана для любой другой операции, то она должна игнорироваться.

Для определения того, какие члены семейства должны выдаваться согласно компоненту **familyReturn** параметра **entryInformationSelection**, вводятся понятия *содействующий член* и *участвующий член*. Эти понятия применимы только к операциям, при которых выдается информация статьи, т. е. к операциям Чтение, Поиск и Модификация статьи.

Если член семейства вносит активный вклад в выполнение операции, то он отмечается как содействующий член. Член семейства вносит вклад в соответствие, если он является частью группировки семейства, которая соответствует фильтру, и если он содержит один или несколько атрибутов, которые сопоставляются неинвертированными элементами фильтра. Он также вносит вклад, если содержит атрибут заданного типа, а инвертированный элемент фильтра для этого типа не дает соответствия. В случае операции Чтение или Модификация статьи член семейства, который выбран операцией (как это определено компонентом операции **object**), является единственным членом, который маркируется как содействующий член и как участвующий член. В случае операции Поиск группировка семейства образуется для сопоставления с фильтром. Если группировка семейства соответствует фильтру (см. п. 7.8.3), то все члены, которые активно способствовали сопоставлению, отмечаются как содействующие члены, в то время как все статьи группировки отмечаются как участвующие члены. Если используемый фильтр – это фильтр "по умолчанию" (**and : { }**), то все члены группировки семейства должны быть отмечены как участвующие члены, а не содействующие члены.

Когда группировка семейства в составной статье соответствует фильтру, а параметр **SearchArgument** определяет выбор иерархии (но не для **self**), выбранные статьи должны также быть отмечены, если это применимо. Если порождающий элемент составной статьи отмечен как участвующий (и, возможно, еще как содействующий), то все упомянутые статьи иерархической группы, которые не являются составными статьями, должны быть выбраны, в остальных случаях они должны быть исключены. Если упомянутая статья – это составная статья, то маркировка ее членов должна быть сделана следующим образом. Каждый член упомянутой составной статьи, который имеет то же самое местное имя члена, как и член составной статьи, соответствующей фильтру, маркируется аналогичным образом. Все другие члены упомянутой составной статьи не маркируются.

Фильтру операции Поиск может соответствовать несколько составных статей, поэтому результирующие выбор и маркировка должны быть объединением выборов и маркировок отдельных составных статей, соответствующих фильтру.

Если статья, соответствующая фильтру, не является составной статьей, ссылающейся на какую-либо составную статью в ее иерархическом выборе, то все члены этой составной статьи отмечаются как участвующие.

Как такая маркировка статей влияет на выдачу информации статьи, детально изложено в п. 7.6.4.

Члены семейства могут быть упакованы в производный атрибут **family-information**. Если выдаваемый результат содержит только один член составной статьи, то упаковка не должна выполняться. Если, однако, несколько членов выдаются после операции Чтение или Модификация статьи, то эти члены должны быть упакованы. В случае операции Поиск, когда выдаются несколько членов составного атрибута, они должны быть упакованы, если не установлена опция управления поиском **separateFamilyMembers**, а если она установлена, то члены должны выдаваться как отдельные статьи.

При выполнении поисковых операций, затрагивающих составные статьи, имеются четыре существенные фазы операции Поиск:

a) Группировки членов семейства в пределах каждой статьи, представляющей интерес, как определено параметром **familyGrouping**, логически рассматриваются в пределах каждой статьи-кандидата (т. е. выбранной подмножеством). Объединяя вместе все атрибуты группы, все значения атрибутов для данного типа атрибутов, можно считать, что они принадлежат к одному типу атрибутов, даже если они были созданы разными членами семейства.

b) Фильтр применяется к каждой группировке семейства; если фильтру удовлетворяет определенная группировка, то составная статья удовлетворяет фильтру и считается выбранной фильтром. Члены семейства маркируются, как описано выше.

c) Маркированные статьи дополняются, как определено компонентом **familyReturn** в параметре **EntryInformationSelection**, маркировкой всех статей, которые должны выдаваться.

d) Если компонент **additionalControl** присутствует в управляющем правиле поиска (см. п. 16.10.8 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то маркировка и, следовательно, то, что выдается, могут быть изменены вследствие обработки упомянутых атрибутов управления.

8 Операции Привязывание и Отвязывание

Операции Привязывание к Справочнику и Отвязывание от Справочника, которые определяются в пп. 8.1 и 8.2 соответственно, используются в DUA в начале и в конце каждого интервала доступа к Справочнику.

8.1 Привязывание к Справочнику

8.1.1 Синтаксис привязывания к Справочнику

Операция Привязывание к Справочнику используется в начале интервала доступа к Справочнику. Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Если запрошено, то Справочник может подписать, зашифровать, либо подписать и зашифровать результаты.

```
directoryBind  OPERATION  ::=  {
        ARGUMENT          DirectoryBindArgument
        RESULT            DirectoryBindResult
        ERRORS            { directoryBindError } }

DirectoryBindArgument  ::=  SET  {
        credentials   [0]     Credentials OPTIONAL,
        versions      [1]     Versions DEFAULT {v1} }

Credentials  ::=  CHOICE  {
        simple                [0]     SimpleCredentials,
        strong                [1]     StrongCredentials,
        externalProcedure     [2]     EXTERNAL,
        spkm                  [3]     SpkmCredentials,
        sasl                  [4]     SaslCredentials }

SimpleCredentials  ::=  SEQUENCE  {
        name          [0]     DistinguishedName,
        validity      [1]     SET {
                time1         [0]     CHOICE {
                        utc             UTCTime,
                        gt              GeneralizedTime } OPTIONAL,
                time2         [1]     CHOICE {
                        utc             UTCTime,
                        gt              GeneralizedTime } OPTIONAL,
                random1       [2]     BIT STRING  OPTIONAL,
                random2       [3]     BIT STRING  OPTIONAL } OPTIONAL,
        password      [2]     CHOICE {
                unprotected           OCTET STRING,
                protected             SIGNATURE {OCTET STRING} } OPTIONAL}

StrongCredentials  ::=  SET  {
        certification-path          [0]     CertificationPath OPTIONAL,
        bind-token                  [1]     Token,
        name                        [2]     DistinguishedName OPTIONAL,
        attributeCertificationPath  [3]     AttributeCertificationPath OPTIONAL }

SpkmCredentials  ::=  CHOICE  {
        req     [0]     SPKM-REQ,
        rep     [1]     SPKM-REP-TI }
```

```
SaslCredentials ::= SEQUENCE {
        mechanism           [0]     DirectoryString { ub-saslMechanism },
        credentials         [1]     OCTET STRING OPTIONAL,
        saslAbort           [2]     BOOLEAN DEFAULT FALSE }
Token ::= SIGNED { SEQUENCE {
        algorithm                   [0]     AlgorithmIdentifier,
        name                        [1]     DistinguishedName,
        time                        [2]     Time,
        random                      [3]     BIT STRING,
        response                    [4]     BIT STRING OPTIONAL,
        bindIntAlgorithm            [5]     SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindIntKeyInfo              [6]     BindKeyInfo OPTIONAL,
        bindConfAlgorithm           [7]     SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindConfKeyInfo             [8]     BindKeyInfo OPTIONAL } }
Versions ::= BIT STRING {v1(0), v2(1) }
DirectoryBindResult ::= DirectoryBindArgument
directoryBindError ERROR ::= {
        PARAMETER               OPTIONALLY-PROTECTED {
                SET {
                        versions            [0]     Versions DEFAULT {v1},
                        error                       CHOICE {
                                serviceError        [1]     ServiceProblem,
                                securityError       [2]     SecurityProblem } } } }
BindKeyInfo ::= ENCRYPTED { BIT STRING }
```

8.1.2 Аргументы привязывания к Справочнику

Аргумент **credentials** (удостоверения) параметра **DirectoryBindArgument** позволяет Справочнику устанавливать подлинность пользователя. Удостоверения могут быть **simple**, **strong** или внешне определенные (**externalProcedure**) (как описано в Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8).

Если используется вариант **simple**, то он состоит из параметра **name** (всегда выделенное имя объекта), необязательного **validity** и необязательного **password**. Это обеспечивает ограниченную степень защиты. Параметр **password** может иметь значение **unprotected** или **protected** (или Protected1, или Protected2), как описано в 18.1 Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8. Параметр **validity** обеспечивает аргументы **time1**, **time2**, **random1** и **random2**, которые получают свои значения согласно двустороннему соглашению и которые могут использоваться для обнаружения подмены. В некоторых образцах защищенный пароль может быть проверен объектом, которому известен пароль, только после местного восстановления защиты для его собственной копии пароля и сравнения результата со значением в привязанном аргументе (**password**). В других образцах может использоваться прямое сравнение.

Компонент **GeneralizedTime** будет использоваться для **time1** и **time2**, если выбранная версия равна **v2** или больше. Использование **GeneralizedTime** при выбранной версии **v1** может препятствовать взаимодействию с реализациями, не имеющими возможности выбора между **UTCTime** и **GeneralizedTime**. Эта ответственность лежит на тех, кто определяет области, в которых эта спецификация Справочника будет использоваться, например, при выделении групп, в которых **GeneralizedTime** может использоваться с определенного времени. Никогда **UTCTime** не будет использоваться для представления дат свыше 2049 года.

Если используется вариант **strong**, то он состоит из параметра **bind-token**, необязательного параметра **certification-path** (сертификата и последовательности перекрестных сертификатов органа сертификации, как определено в Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8) и параметра **name** инициатора. Это дает возможность Справочнику подтвердить подлинность инициатора, устанавливающего ассоциацию, и наоборот. Если в операции привязывания используется параметр **StrongCredentials** или **SpkmCredentials**, то передается информация, касающаяся подлинности и автонизации. Это дает возможность подтвердить подлинность любого объекта, а также дает возможность использовать установленные криптографические ключевые данные для шифрования и защиты целостности.

Компоненты **bindIntAlgorithm** и **bindConfAlgorithm** применяются, чтобы выбрать криптографические алгоритмы, используемые для защиты последующих операций привязывании. Инициатор запроса включает список поддерживаемых алгоритмов в порядке предпочтения. Справочник выбирает из списка один алгоритм, который соответствует его собственной стратегии защиты, и указывает его в ответе.

Ключи сеанса связи, которые должны использоваться алгоритмами защиты целостности и конфиденциальности, устанавливаются с помощью полей **bindIntKeyInfo** и **bindConfKeyInfo**. Как инициатор, так и Справочник, могут вносить вклад в выбор ключа сеанса, генерируя ключ сеанса соответствующей длины и зашифровав его другим открытым ключом. Ключ сеанса – это ИСКЛЮЧАЮЩЕЕ ИЛИ таких двух компонентов. Следует отметить, что инициатор может разрешить генерацию ключа сеанса Справочнику, тогда вышеупомянутые поля будут опущены в привязывающем аргументе.

ПРИМЕЧАНИЕ 1. – Удостоверения, требуемые для аутентификации, могут переноситься в элементе Security Exchange Service Element (см. Рекомендацию МСЭ-Т X.519 | ИСО/МЭК 9594-5), в этом случае они отсутствуют в аргументах и результатах привязывания.

Если операция должна быть подписана и зашифрована, то сертификат атрибута, содержащий этот атрибут (см. раздел 12 Рекомендации МСЭ-Т X.509 | ИСО/МЭК 9594-8), может использоваться для переноса свидетельств, необходимых для доступа к этому атрибуту. Параметр **attributeCertificationPath** используется для переноса свидетельства безопасности для управления доступом на основе правила или другого атрибута, переданного в Сертификате атрибута, возможно, с сертификатами, необходимыми для проверки действительности Сертификата атрибута.

Аргументы маркера привязывания используются следующим образом: параметр **algorithm** – это идентификатор алгоритма, используемого для подписи этой информации, параметр **name** – это имя предполагаемого получателя. Параметр **time** содержит интервал действия маркера. Число **random** – это число, которое должно быть различно для каждого еще действующего маркера; оно может использоваться получателем для обнаружения атак подмены.

ПРИМЕЧАНИЕ 2. – Там, где имена используются в простом или сильном удостоверении, можно использовать альтернативные выделенные имена, если они существуют. Однако аутентификация и управление доступом, основанные на имени, не могут работать так, как необходимо, если используется не первичное выделенное имя. После успешной обработки аутентифицированной операции Привязывание, при любом имени в аргументе Привязывание, связанные объекты должны узнать друг друга по их первичным выделенным именам, чтобы облегчить работу параметров управления доступом во время операции Привязывание.

Если используется **externalProcedure**, то семантика используемой схемы аутентификации не входит в эти спецификации Справочника.

sasl применяется при использовании уровня простой аутентификации и безопасности (SASL), определенного в RFC 2222. Если операция **directoryBind** вызывается со значением механизма **SaslCredentials**, установленным на пустую строку, то должны выдаваться **SecurityError** в **inappropriateAuthentication**.

Аргумент **versions** в параметре **DirectoryBindArgument** указывает версии службы, к которой подготовлен DUA. Значение **v1** обозначает версию 1 протокола, а значение **v2** обозначает версию 2 протокола. Значение **v2** должно использоваться, если в последующей операции **ModifyEntry** значения **alterValues** или **resetValue** типов модификации должны быть переданы в запросе или запрашивается результат, отличный от **NULL** (см. 11.3). Значение должно быть установлено в **v2**, если используется подписывание ошибок или результатов для Добавления статьи, Удаления статьи, Модификации статьи, Модификации DN.

Переход к будущим версиям Справочника будет облегчаться следующим:

a) любые элементы **DirectoryBindArgument**, отличающиеся от тех, которые определены в этой спецификации Справочника, должны приниматься и игнорироваться;

b) не имеющие определения дополнительные варианты именованных битов в **DirectoryBindArgument** (например, версии) должны приниматься и игнорироваться.

Компонент **response** используется для переноса случайного числа, если требуется опознавательный ответ при аутентификации.

Компоненты **bindIntAlgorithm**, **bindKeyInfo**, **bindConfAlgorithm** и **bindConfKey** используются для переноса информации, применяемой для защиты последующих операций привязывания.

8.1.3 Результаты привязывания к Справочнику

Если запрос привязывания будет успешным, то должен быть обратно передан результат.

Аргумент **credentials**, входящий в **DirectoryBindResult**, позволяет пользователю установить подлинность Справочника. Он позволяет передавать в DUA информацию, идентифицирующую DSA (что непосредственно обеспечивается службой Справочника). Он должен иметь ту же форму (т. е. **CHOICE**), которая представлена пользователем.

Параметр **versions** в **DirectoryBindResult** указывает, какую из версий службы, запрошенных DUA, фактически готов обеспечить DSA.

8.1.4 Ошибки привязывания к Справочнику

Если запрос привязывания заканчивается неуспешно, то должно быть обратно передано уведомление об ошибке привязывания.

Параметр **versions** в **DirectoryBindError** указывает, какие версии поддерживает DSA.

Параметры **securityError** или **serviceError** представляются следующим образом:

- **securityError inappropriateAuthentication**
  invalidCredentials
  blockedCredentials
- **serviceError unavailable**
  saslBindInProgress

8.2 Отвязывание от Справочника

Отвязывание в конце интервала доступа к Справочнику существует для среды OSI, определенной в пп. 7.6.4 и 7.6.5 Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5 и для среды TCP/IP, определенной в п. 9.3.2 Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5.

ПРИМЕЧАНИЕ. – В течение отвязывания все результаты в виде страниц, к которым более не обращаются, становятся недоступными и должны быть удалены.

9 Операции чтения из Справочника

Существуют две операции, "подобные чтению": **read** (чтение) и **compare** (сравнение), которые определяются в пп. 9.1 и 9.2 соответственно. Операция **abandon** (отказ), определенная в 9.3, присоединена к этим операциям для удобства.

9.1 Чтение

9.1.1 Синтаксис чтения

Операция Чтение используется для получения информации из явно идентифицированной статьи. Она может также использоваться для проверки выделенного имени. Аргументы операции могут быть подписаны инициатором запроса (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Если запрошено, то Справочник может подписать результат.

```
read OPERATION ::= {
    ARGUMENT        ReadArgument
    RESULT          ReadResult
    ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                    securityError }
    CODE            id-opcode-read}

ReadArgument ::= OPTIONALLY-PROTECTED {
    SET{
        object                  [0]  Name,
        selection               [1]  EntryInformationSelection DEFAULT {},
        modifyRightsRequest     [2]  BOOLEAN DEFAULT FALSE,
        COMPONENTS OF                CommonArguments } }

ReadResult ::= OPTIONALLY-PROTECTED {
    SET{
        entry               [0]  EntryInformation,
        modifyRights        [1]  ModifyRights OPTIONAL,
        COMPONENTS OF            CommonResults } }

ModifyRights ::= SET OF SEQUENCE {
    item                    CHOICE {
            entry           [0]  NULL,
            attribute       [1]  AttributeType,
            value           [2]  AttributeValueAssertion},
    permission              [3]  BIT STRING { add (0), remove (1), rename (2), move (3) } }
```

9.1.2 Аргументы чтения

Аргумент **object** указывает статью объекта, из которой запрашивается информация. Если значение **Name** содержит один или более псевдонимов, то они переименовываются (если это не запрещено соответствующими параметрами управления службой). Значение **Name** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **selection** указывает, какая информация из статьи запрашивается (см. п. 7.6). Однако не следует предполагать, что выдаваемые атрибуты будут теми же, что и запрошенные, либо будут ограничиваться запрошенными.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Для целей данной операции компонент **sizeLimit** не нужен, а если он предоставляется, то игнорируется. Если аргумент данной операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором запроса, то компонент **SecurityParameters** (см. п. 7.10) включается в эти аргументы.

Аргумент **modifyRightsRequest** используется для запроса ответного сообщения, содержащего информацию о правах инициатора на модификацию статьи и ее атрибутов.

9.1.3 Результаты чтения

Если запрос будет успешным, то должен быть обратно передан результат.

Параметр результата о статье содержит требуемую информацию (см. п. 7.7). Он может содержать информацию семейства, если она запрошена присутствием элемента **familyReturn** в **EntryInformationSelection**.

Параметр **modifyRights** присутствует, если это было запрошено аргументом **modifyRightsRequest**, и сообщает, что пользователь имеет привилегии на модификацию части или всей запрашиваемой информации из статьи, а возвращение этой информации разрешается местной стратегией безопасности. Если информация возвращается, то права инициатора на модификацию возвращаются для статьи и для атрибутов, указанных в аргументе **selection**. Этот параметр содержит следующие элементы:

- Какой-либо элемент из **SET** возвращается для **entry**; для каждого запрошенного пользователем **attribute**, на который пользователь имеет право добавления или удаления; и для каждого возвращаемого **value** атрибута, для которого права пользователя на добавление или удаление отличаются от прав в соответствующем атрибуте.
- Возвращенный параметр **permission** указывает на операции или действия пользователя со статьей, которые были успешными. В случае статьи значение **remove** указывает, что операция **RemoveEntry** была успешной; значение **rename** указывает, что операция **ModifyDN** с отсутствием параметра **newSuperior** была успешной; а значение **move** указывает, что операция **ModifyDN** с присутствием параметра **newSuperior** и неизмененным RDN была успешной.

  В случае атрибутов и значений **add** указывает, что операция **ModifyEntry**, которая добавляет атрибут или значение, была успешной; а значение **remove** указывает, что операция **ModifyEntry**, которая удаляет атрибут или значение, была успешной.

ПРИМЕЧАНИЕ. – Операция move (перемещение) статьи в новую старшую статью может также зависеть от разрешений, связанных с этой новой старшей статьей (например, от **basic-access-control**). Если **permission** определяется, то это условие игнорируется.

Информация **CommonResults** (см. п. 7.4) содержит параметры безопасности, применяемые к ответу. Если результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) включается в результаты.

9.1.4 Ошибки чтения

Если запрос завершился неудачей, то будет выдано одно из перечисленных уведомлений об ошибке. Если ни один из атрибутов, явно перечисленных в аргументе **selection**, не может быть выдан, то должно быть выдано уведомление **attributeError** с указанием проблемы **noSuchAttributeOrValue**. Обстоятельства, при которых должны выдаваться другие уведомления об ошибке, определяются в разделе 12.

9.1.5 Точки принятия решения операции Чтение при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **basic-access-control** влияет на читаемую статью, то применяется следующая последовательность управлений доступом:

1) Разрешение *Read* требуется для прочтения статьи. Если разрешение не предоставляется, то операция получает отказ согласно п. 7.11.1.3.

2) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться только типы атрибута, то для каждого типа атрибута, который должен выдаваться, требуется разрешение *Read*. Если разрешение не предоставляется, то такой тип атрибута не включается в **ReadResult**. Если вследствие применения этих параметров управления никакая информация атрибутов не выдается, то вся операция оканчивается неудачей согласно п. 9.1.5.1.

3) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться типы и значения атрибутов, то для каждого типа атрибута и для каждого значения, которые должны выдаваться, требуется разрешение *Read*. Если разрешение для типа атрибута не предоставляется, то такой атрибут не включается в **ReadResult**. Если разрешение для значения атрибута не предоставляется, то это значение опускается в соответствующем атрибуте. Если разрешение не предоставляется любому из значений атрибута, то выдается элемент **Attribute**, содержащий пустое множество **SET OF AttributeValue**. Если вследствие применения этих параметров управления никакая информация атрибутов не выдается, то вся операция оканчивается неудачей согласно п. 9.1.5.1.

ПРИМЕЧАНИЕ. – Привилегии, разрешающие операцию DAP, могут не работать в среде DAP, где для поддержки эквивалентной услуги чтения требуется разрешение на просмотр.

9.1.5.1 Выдача уведомлений об ошибках

Если операция оканчивается неудачей согласно пунктам 2) или 3) подраздела 9.1.5, то действительным уведомлением об ошибке является одно из следующих:

a) Если была указана открытая опция (т. е. параметры **allUserAttributes** или **allOperationalAttributes**), то выдается параметр **securityError** с указанием причины **insufficientAccessRights** или **noInformation**.

b) В противном случае, если была указана опция **select** (в параметрах **attribute** и/или **extraAttributes**), то при предоставленном разрешении *DiscloseOnError* для любого из выбранных атрибутов выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**. В других случаях выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

9.1.5.2 Нераскрытие неполных результатов

Если неполный результат выдается в **EntryInformation**, т. е. некоторые из атрибутов или значений атрибута были опущены из-за применения соответствующих параметров управления доступом, то элемент **incompleteEntry** должен быть установлен в **TRUE**, если разрешение *DiscloseOnError* предоставляется, по крайней мере, для одного типа атрибута, отсутствующего в результате, или, по крайней мере, для одного значения атрибута, отсутствующего в результате (для типа атрибута, на который было предоставлено разрешение *Read*).

9.1.6 Точки принятия решения операции Чтение при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на читаемую статью, то применяется следующее управление доступом:

1) Если доступ на уровне статьи отклонен процедурой **rule-based-access-control**, то операция заканчивается неудачей с параметром **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) Если доступ к статье не разрешается согласно схеме **basic-access-control**, как описано в пункте 1) подраздела 9.1.5, то операция заканчивается неудачей согласно п. 7.11.1.3.

3) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться только типы атрибута, а процедурой **rule-based-access-control** доступ не предоставляется для всех значений атрибута определенного типа, то такой тип атрибута не включается в **ReadResult**. Если вследствие применения этих параметров управления никакая информация атрибутов не выдается, то вся операция оканчивается неудачей с выдачей **attributeError** с указанием причины **noSuchAttributeOrValue** согласно п. 9.1.5.1 b).

4) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться только типы атрибута, то применяется процедура **basic-access-control**, как описано в п. 9.1.5 2).

5) При управлении доступом на основе правила, если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться типы и значения атрибутов, то для каждого значения атрибута, который должен выдаваться, должен быть разрешен доступ. Если доступ к какому-либо значению атрибута не разрешен, то это значение атрибута опускается в соответствующем атрибуте. Если доступ не разрешается любому из значений атрибута, то весь атрибут опускается в **ReadResult**. Если вследствие применения этих параметров управления никакая информация атрибутов не выдается, то вся операция оканчивается неудачей с выдачей **attributeError** с указанием причины **noSuchAttributeOrValue**.

6) Процедура **basic-access-control** применяется согласно п. 9.1.5, пункт 3).

7) Имя статьи, выдаваемое в результате операции, определено в п. 7.11.2.2.

9.2 Сравнение

9.2.1 Синтаксис сравнения

Операция Сравнение используется для сравнения некоторого значения (которое предоставляется как аргумент запроса) со значением(ями) атрибута конкретного типа в конкретной статье объекта. Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором операции (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Если запрошено, то Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

Может использоваться любое значение **familyGrouping**, кроме **multiStrand**, а атрибуты во всех сгруппированных членах семейства должны использоваться при сравнении с утверждением о предполагаемом значении атрибута. Если **familyGrouping** указывает **multiStrand**, то допускается использование **compoundEntry**.

```
compare OPERATION ::= {
        ARGUMENT    CompareArgument
        RESULT      CompareResult
        ERRORS      { attributeError | nameError | serviceError | referral | abandoned |
                    securityError }
        CODE        id-opcode-compare}

CompareArgument ::= OPTIONALLY-PROTECTED {
        SET{
            object              [0]  Name,
            purported           [1]  AttributeValueAssertion,
            COMPONENTS OF            CommonArguments } }

CompareResult ::= OPTIONALLY-PROTECTED {
        SET{
            name                Name OPTIONAL,
            matched             [0]  BOOLEAN,
            fromEntry           [1]  BOOLEAN DEFAULT TRUE,
            matchedSubtype      [2]  AttributeType OPTIONAL,
            COMPONENTS OF       CommonResults } }
```

9.2.2 Аргументы сравнения

Аргумент **object** – это имя конкретной рассматриваемой статьи объекта. Если **Name** содержит один или более псевдонимов, то они переименовываются (если это не запрещено соответствующим параметром управления службы). **Name** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **purported** указывает тип и значение атрибута для сравнения с таковым в статье. Сравнение дает TRUE, если статья содержит предполагаемый тип атрибута или один из его подтипов, или имеется совокупный атрибут статьи, который является предполагаемым типом атрибута или одним из его подтипов (см. п. 7.6), и если имеется значение такого атрибута, которое соответствует предполагаемому значению, при использовании правила сопоставления **equality** атрибута.

ПРИМЕЧАНИЕ. – Запрос **compare** не может быть выполнен типом атрибута "друга" типа атрибута, определенного в аргументе.

Если в определение значения атрибута включены контекстные утверждения, то сопоставление должно выполняться только для тех значений, которые удовлетворяют всем данным контекстным утверждениям, как описано в п. 8.9.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если контекстные утверждения не включены в определение значения атрибута, то должны применяться контекстные определения "по умолчанию", как описано в п. 8.9.2.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Для целей данной операции компонент **sizeLimit** не нужен, а если он предоставляется, то игнорируется. Если аргумент данной операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в эти аргументы.

9.2.3 Результаты сравнения

Если запрос будет успешным (т. е. сравнение фактически выполнено), то должен быть обратно передан результат.

Параметр **name** – это выделенное имя статьи или имя псевдонима статьи, описанное в п. 7.7. Оно присутствует только тогда, когда псевдоним был переименован, RDN были приведены к первичным RDN, либо применен контекстный выбор, а выдаваемое имя отличается от имени **object**, указанного в аргументе операции.

Параметр результата **matched** содержит результат сравнения. Этот параметр принимает значение **TRUE**, если значения были сравнены и они соответствуют, либо принимают значение **FALSE**, если нет.

Если параметр **fromEntry** имеет значение **TRUE**, то информация сравнивалась со статьей; если имеет значение **FALSE**, то информация сравнивалась с копией.

Параметр **matchedSubtype** присутствует только в случае, когда результат сопоставления равен TRUE, а сопоставление прошло успешно благодаря тому, что соответствовал подтип предполагаемого атрибута. Этот параметр содержит подтип, который соответствовал. Если имеется несколько таких подтипов, то выдается один с самым высоким уровнем в иерархии.

Информация **CommonResults** (см. п. 7.4) содержит параметры безопасности, применимые к ответу. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) включается в результаты.

9.2.4 Ошибки сравнения

Если запрос завершился неудачей, то будет выдано одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны выдаваться уведомления о конкретной ошибке, определяются в разделе 12.

9.2.5 Точки принятия решения операции Сравнение при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **basic-access-control** влияет на сравниваемую статью, то применяется следующая последовательность управления доступом:

1) Разрешение *Read* требуется для сравнения статьи. Если разрешение не предоставляется, то операция получает отказ согласно п. 7.11.1.3.

2) Разрешение *Compare* требуется для сравниваемого атрибута. Если разрешение не предоставляется, то операция получает отказ согласно п. 9.2.5.1.

3) Если существует значение сравниваемого атрибута, которое соответствует аргументу **purported** и для которого предоставлено разрешение *Compare*, то операция выдает значение **TRUE** в компоненте **matched** параметра результата **CompareResult.** В противном случае операция выдает значение **FALSE**.

9.2.5.1 Выдача уведомлений об ошибке

Если операция оканчивается неудачей согласно пункту 2) подраздела 9.2.5, то действительным уведомлением об ошибке является одно из следующих: если разрешение *DiscloseOnError* предоставляется сравниваемому атрибуту, то выдается параметр **securityError** с указанием причины **insufficientAccessRights** или **noInformation**. В других случаях выдается параметр **attributeError** с указанием причины **noSuchAttributeOrValue**.

9.2.6 Точки принятия решения операции Сравнения при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на сравниваемую статью, то применяется следующее управление доступом:

1) если доступ на уровне статьи отклонен процедурой **rule-based-access-control**, то операция оканчивается неудачей с параметром **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4;

2) если доступ к статье не разрешается согласно схеме **basic-access-control**, как описано в пункте 1) подраздела 9.2.5, то операция оканчивается неудачей согласно п. 7.11.1.3;

3) если доступ не разрешен для сравниваемого значения атрибута, то Справочник должен действовать так, как будто этого значения атрибута нет;

4) процедура **basic-access-control** применяется согласно пунктам 2) и 3) подраздела 9.2.5;

5) имя, выдаваемое в результате операции, определено в п. 7.11.2.2.

9.3 Отказ

Операции, при которых запрашивается Справочник, могут отменяться с использованием операции **abandon**, если пользователь потерял интерес к результату. Параметры этой операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Если запрошено, то Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
abandon OPERATION  ::= {
        ARGUMENT        AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE            id-opcode-abandon}

AbandonArgument ::= OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            invokeID            [0]     InvokeId}}

AbandonResult ::= CHOICE {
        null                NULL,
        information         OPTIONALLY-PROTECTED-SEQ {
                                SEQUENCE {
                                    invokeID                InvokeId,
                                    COMPONENTS OF           CommonResultsSeq } } }
```

Существует единственный аргумент **invokeID** для указания операции, которая должна быть отменена. Значение **invokeID** будет таким же, как и значение **invokeID**, которое использовалось для вызова операции, которая должна быть отменена.

Если запрос будет успешным, то должен быть обратно передан некоторый результат. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) параметра **CommonResultsSeq** (см. п. 7.4) включается в результат. Если результат операции не должен подписываться Справочником, то никакая информация не передается в результате. Первоначальная операция должна окончиться неудачей с уведомлением об ошибке **abandoned**.

Если запрос оказался безуспешным, то должно быть выдано уведомление об ошибке **abandonFailed**. DSA может, согласно местному решению, не отказываться от операции и затем выдать уведомление об ошибке **abandonFailed**. Это уведомление об ошибке описано в п. 12.3.

Отказ применим только к запрашивающим операциям, т. е. к операциям Чтение, Сравнение, Список и Поиск.

DSA может на месте прекратить выполнение какой-либо операции. Если DSA был сцеплен или получил многоадресную связь для операции с другими DSA, то он, в свою очередь, может запросить их об отказе от этой операции.

10 Операции поиска в Справочнике

Имеются две операции, "подобные поиску": Список и Поиск, которые определяются в пп. 10.1 и 10.2 соответственно.

10.1 Список

10.1.1 Синтаксис операции Список

Операция Список используется, чтобы получить список статей, непосредственно подчиненных явно идентифицированной статье. При некоторых обстоятельствах выдаваемый список может быть неполным. Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Если запрошено, то Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
list    OPERATION  ::= {
        ARGUMENT        ListArgument
        RESULT          ListResult
        ERRORS          { nameError | serviceError | referral | abandoned | securityError }
        CODE            id-opcode-list }

ListArgument  ::=  OPTIONALLY-PROTECTED  {
        SET {
            object                  [0]     Name,
            pagedResults            [1]     PagedResultsRequest OPTIONAL,
            listFamily              [2]     BOOLEAN DEFAULT FALSE,
            COMPONENTS OF                   CommonArguments } }
```

```
ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
                listInfo                        SET {
                        name                                    Name  OPTIONAL,
                        subordinates            [1]     SET OF SEQUENCE {
                                rdn                                     RelativeDistinguishedName,
                                aliasEntry                      [0]     BOOLEAN DEFAULT  FALSE,
                                fromEntry                       [1]     BOOLEAN DEFAULT  TRUE },
                        partialOutcomeQualifier [2]     PartialOutcomeQualifier OPTIONAL,
                        COMPONENTS OF                   CommonResults },
                uncorrelatedListInfo    [0]     SET OF ListResult } }

PartialOutcomeQualifier ::= SET {
        limitProblem                    [0]     LimitProblem  OPTIONAL,
        unexplored                      [1]     SET SIZE (1..MAX) OF ContinuationReference  OPTIONAL,
        unavailableCriticalExtensions   [2]     BOOLEAN DEFAULT  FALSE,
        unknownErrors                           [3]     SET     SIZE    (1..MAX)    OF    ABSTRACT-SYNTAX.&Type
OPTIONAL,
        queryReference                  [4]     OCTET STRING  OPTIONAL,
        overspecFilter                  [5]     Filter OPTIONAL,
        notification                    [6]     SEQUENCE SIZE (1..MAX) OF Attribute  OPTIONAL,
        entryCount                              CHOICE {
                bestEstimate            [7]     INTEGER,
                lowEstimate             [8]     INTEGER,
                exact                   [9]     INTEGER } OPTIONAL,
        streamedResult                  [10]    BOOLEAN DEFAULT  FALSE }

LimitProblem ::= INTEGER {
        timeLimitExceeded (0), sizeLimitExceeded (1), administrativeLimitExceeded (2) }
```

10.1.2 Аргументы операции Список

Аргумент **object** идентифицирует статью объекта (или, возможно, корень), непосредственно подчиненные статьи которой должны быть перечислены. Если **Name** содержит один или более псевдонимов, то они переименовываются (если это не запрещено соответствующим параметром управления службой). **Name** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **pagedResults** используется для того, чтобы запросить результаты операции в режиме постраничного вывода, как описано в п. 7.9.

Если значение аргумента **listFamily** равно **TRUE**, а аргумент **object** является порождающим элементом, то перечисленные подчиненные статьи берутся из непосредственно подчиненных членов семейства; никакие другие подчиненные статьи не включаются. В противном случае перечисленные подчиненные статьи берутся только из непосредственно подчиненных статей, которые не являются членами семейства.

Информация **CommonArguments** (см. п. 7.3) содержит спецификации параметров управления службой, применяемые к запросу. Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

10.1.3 Результаты операции Список

Запрос завершается успешно согласно параметрам управления доступом, если **object** найден, независимо от того, имеется ли подчиненная информация для выдачи в ответе.

Параметр **name** – это выделенное имя статьи или имя псевдонима статьи, описанные в п. 7.7. Он присутствует только в случае, когда псевдоним был переименован, RDN были приведены к первичным RDN, либо был применен контекстный выбор, а имя, которое должно выдаваться, отличается от имени **object**, указанного в аргументе операции.

Параметр **subordinates** переносит информацию о непосредственно подчиненных статьях, если они имеются для названной статьи. Если какие-либо подчиненные статьи являются псевдонимами, то они не должны быть переименованы.

Параметр **rdn** – это относительно выделенное имя подчиненной статьи. На него может повлиять контекст, как описано для **Name** в п. 7.7.

Параметр **fromEntry** указывает, что информация была получена от статьи (**TRUE**) или из копии статьи (**FALSE**).

Параметр **aliasEntry** указывает, является ли подчиненная статья статьей псевдонима (**TRUE**) или нет (**FALSE**).

Параметр **PartialOutcomeQualifier** состоит из девяти подкомпонентов, рассматриваемых ниже. Этот параметр будет присутствовать в случае, когда результат является неполным из-за ограничения времени, ограничения размера или ограничений административного характера, либо потому, что области DIT не были исследованы, потому что некоторые критические расширения были недоступны, потому что была получена неизвестная ошибка, потому что результаты выдаются постранично, потому что должен быть указан переопределенный фильтр, потому что должны быть выданы один или большее количество атрибутов уведомления, или потому что результатом операции является результат, направленный в потоке, и этот ответ не является последним ответом результата:

a) Параметр **limitProblem** указывает, были ли превышены ограничения времени, ограничения размера или ограничения административного характера. Выдаваемый результат будет содержать ту информацию, которая была доступна при достижении ограничения.

b) Параметр **unexplored** будет присутствовать, если области DIT не исследовались. Содержащаяся в параметре информация позволяет DUA продолжить операцию Список путем обращения к другим пунктам доступа, если он выберет этот способ. Параметр состоит из набора (возможно, пустого) параметров **ContinuationReferences**, каждый из которых состоит из имени основного объекта, от которого операция может быть продолжена, соответствующего значения **OperationProgress** и набора пунктов доступа, от которых может быть продолжена обработка запроса. Выдаваемые **ContinuationReferences** должны быть в пределах области отсылок, запрошенной при операции параметром управления службой. См. п. 12.6.

c) Параметр **unavailableCriticalExtensions**, если присутствует, указывает, что одно или более критических расширений были недоступны в некоторой части Справочника.

d) Параметр **unknownErrors** используется для выдачи уведомления о неизвестных типах ошибок или параметров, полученных от других DSA при выполнении операции. Каждый член в SET содержит сведения об одной такой неизвестной ошибке. См. п. 12.2.4 Рекомендации МСЭ-Т X.519 | ИСО/МЭК 9594-5.

e) Параметр **queryReference** должен присутствовать, если DUA затребовал результаты в виде страниц, а DSA не выдал все доступные результаты. См. п. 7.9. Параметр отсутствует, если DSA может определить, что все результаты, действительные для пользователя, были выданы (т.е. кроме результата применения управления доступом).

f) Компонент **overspecFilter** используется только вместе с операцией Поиск, когда из-за узко заданного фильтра выдан пустой результат операции Поиск, хотя имеются статьи-кандидаты, соответствующие только части фильтра или только приблизительно соответствующие фильтру. Он выдается только тогда, когда запрос поиска содержит пункт **checkOverspecified**, и Справочник может определить, что фильтр был узко задан. Он содержит фильтр, указанный в аргументе **search**, с теми элементами фильтра, которые дали соответствие для некоторых не включенных статей. Фактическая процедура для создания компонента **overspecFilter** является местным вопросом.

   ПРИМЕЧАНИЕ 1. – Выдача подходящего **overspecFilter** в распределенном Справочнике является предметом для дальнейшего изучения.

g) Параметр **notification** может использоваться, чтобы выдавать описание ошибки, и может также использоваться в операции Поиск, чтобы выдать атрибут **proposedRelaxation** (см. п. 5.12.15 Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6), определяющий стратегию ослабления, которая могла бы применяться пользователем. В этом случае может быть указана последовательность элементов **MRMapping**, которая могла бы использоваться для воздействия на стратегию ослабления (или ужесточения), определенную соответствующим правилом поиска.

   ПРИМЕЧАНИЕ 2. – Порядок следования атрибутов в **notification** не имеет значения.

h) Параметр **entryCount** относится только к результатам поиска, и если он присутствует, то дает лучшую оценку числа статей, которые удовлетворяют критериям поиска. Этот подкомпонент должен присутствовать тогда и только тогда, когда:

   – опция управления поиском **entryCount** установлена в аргументе поиска или в управляющем правилепоиска,

   – были запрошены результаты в виде страниц или ограничение размера было превышено, и

   – это свойство поддержано, по крайней мере, одним из участвующих DSA.

   Когда присутствует подкомпонент **entryCount**, берется **bestEstimate** или должен быть сделан правильный выбор, если все участвующие в операции DSA поддерживают это свойство, и если все удовлетворяющие требованиям DSA участвовали в операции. Правильный выбор делается, если все участвующие агенты DSA могут дать точный отсчет, в противном случае выбирается **bestEstimate**. Если не все удовлетворяющие требованиям DSA участвовали в операции или если некоторые из участвующих DSA не поддерживают параметр **entryCount**, делается выбор **lowEstimate**. Члены семейства составной статьи считаются как одиночная статья.

i) Если имеется параметр **streamedResult** со значением TRUE, то он указывает на то, что DSA направляет результат в потоке и что этот ответ не является последним ответом результата. Если этот параметр отсутствует или имеет значение FALSE, то он указывает на то, что данный ответ является окончательным ответом направленного в потоке результата или что он является ответом, не направленным в потоке. Каждый из ответов в результате, направленном в потоке, определяется с тем же **invokeId**.

Если возникает проблема ограничения, которая приводит к тому, что элемент **limitProblem** используется в **PartialOutcomeQualifier**, этот компонент должен быть повторен во всех последующих результатах как часть набора результатов в виде страниц.

ПРИМЕЧАНИЕ 3. – Каждый из ответов в результате, направленном в потоке, определяется с тем же **invokeID**. Сам по себе такой вариант имеется вместе с протоколами Справочника IDM, указанными в Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5.

Если DUA выдал запрос на защиту путем подписи или если Справочник по другим причинам не способен увязать информацию, то параметр **uncorrelatedListInfo** может содержать ряд наборов параметров результата, созданных и подписанных различными компонентами Справочника. Если никакой DSA, участвующий в сцеплении, не может увязать все результаты, то DUA должен объединить фактический результат от различных частей.

Информация **CommonResults** (см. п. 7.4) содержит параметры безопасности, применимые к ответу. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) включается в результаты.

10.1.4 Ошибки операции Список

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

10.1.5 Точки принятия решения операции Списка при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если **basic-access-control** влияет на часть DIB, в которой выполняется операция Список, то применяется следующая последовательность параметров управления доступом:

1) Для статьи, указанной аргументом **object**, не требуется никакого специального разрешения.

2) Для каждой непосредственно подчиненной статьи, для которой параметр **RelativeDistinguishedName** должен выдаваться в **subordinates**, требуются разрешения *Browse (Просмотр)* и *ReturnDN (Выдача DN)*. Статьи, для которых эти разрешения не даны, игнорируются. Если вследствие применения этих параметров управления никакая информация подчиненной статьи (кроме **ContinuationReferences** в **PartialOutcomeQualifier**) не выдается и если разрешение *DiscloseOnError* не дано для статьи, указанной аргументом **object**, то операция оканчивается неудачей, и выдается **nameError** с указанием причины **noSuchObject**. Элемент **matched** должен содержать или имя следующей старшей статьи, для которой разрешение *DiscloseOnError* дано, или имя корня DIT (т. е. пустой параметр **RDNSequence**). В противном случае операция выполняется успешно, но никакая информация подчиненной статьи (кроме **ContinuationReferences** в **PartialOutcomeQualifier**) не переносится в ответе.

ПРИМЕЧАНИЕ 1. – В случае выдаваемого параметра **nameError** пустая последовательность **RDNSequence** может использоваться в DSA, который не имеет доступа ко всем старшим статьям.

ПРИМЕЧАНИЕ 2. – Стратегия безопасности может предотвращать раскрытие подчиненной информации, которая иначе была бы передана как **ContinuationReferences** в **PartialOutcomeQualifier**. Если такая стратегия действует и если DUA ограничивает службу, указывая **chainingProhibited**, то Справочник может выдать уведомление **serviceError** с указанием причины **chainingRequired**. В противном случае применяется процедура, описанная выше в пункте 2).

ПРИМЕЧАНИЕ 3. – Стратегия безопасности может удерживать Справочник от указания на то, что подчиненная статья, занесенная в список, является статьей псевдонима. Например, если для DUA нет разрешения *Read* для доступа к статье псевдонима, к содержащимся там атрибуту **objectClass** и значению **alias**, то Справочник может опустить компонент **aliasEntry** в **subordinates** из **ListResult** или установить его в значение **FALSE**.

ПРИМЕЧАНИЕ 4. – Если разрешение *DiscloseOnError* не предоставлено статье, указанной аргументом **object**, то параметр **PartialOutcomeQualifier** с указанием **limitProblem** или **unavailableCriticalExtensions** не должен выдаваться, так как это может нарушить безопасность данной статьи.

10.1.6 Точки принятия решения операции Список при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control,** или **rule-and-simple-access-control** влияет на часть DIB, в которой выполняется операция Список, то применяется следующее управление доступом:

1) Если разрешение на уровне статьи на основе правила отклонено для статьи, указанной аргументом **object**, то выдается **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) Для каждой непосредственно подчиненной статьи, для которой параметр **RelativeDistinguishedName** должен выдаваться в **subordinates**, должно быть предоставлено разрешение на RDN на основе правила. Статьи, для которых доступ не разрешен, игнорируются.

3) Процедура **basic-access-control** применяется, как описано в п. 10.1.5.

10.2 Поиск

10.2.1 Синтаксис поиска

Операция Поиск используется для поиска в одной или нескольких частях Справочника статей, представляющих интерес, и для выдачи выбранной информации из этих статей. Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Когда затребовано, Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
search OPERATION ::= {
        ARGUMENT        SearchArgument
        RESULT          SearchResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-search }

SearchArgument ::= OPTIONALLY-PROTECTED {
        SET{
            baseObject                  [0]     Name,
            subset                      [1]     INTEGER {
                baseObject(O), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
            filter                      [2]     Filter DEFAULT and : { },
            searchAliases               [3]     BOOLEAN DEFAULT TRUE,
            selection                   [4]     EntryInformationSelection DEFAULT { },
            pagedResults                [5]     PagedResultsRequest OPTIONAL,
            matchedValuesOnly           [6]     BOOLEAN DEFAULT FALSE,
            extendedFilter              [7]     Filter OPTIONAL,
            checkOverspecified          [8]     BOOLEAN DEFAULT FALSE,
            relaxation                  [9]     RelaxationPolicy  OPTIONAL,
            extendedArea                [10]    INTEGER   OPTIONAL,
            hierarchySelections         [11]    HierarchySelections DEFAULT { self },
            searchControlOptions        [12]    SearchControlOptions DEFAULT { searchAliases },
            joinArguments               [13]    SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
            joinType                    [14]    ENUMERATED {
                innerJoin(O), leftOuterJoin(1), fullOuterJoin(2) } DEFAULT leftOuterJoin,
            COMPONENTS OF                       CommonArguments } }

HierarchySelections ::= BIT STRING {
        self            (0),
        children        (1),
        parent          (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
        siblingChildren (7),
        siblingSubtree  (8),
        all             (9)}
```

```
SearchControlOptions ::= BIT STRING {
        searchAliases               (0),
        matchedValuesOnly           (1),
        checkOverspecified          (2),
        performExactly              (3),
        includeAllAreas             (4),
        noSystemReiaxation          (5),
        dnAttribute                 (6),
        matchOnResidualName         (7),
        entryCount                  (8),
        useSubset                   (9),
        separateFamilyMembers       (10),
        searchFamily                (11)}

JoinArgument ::= SEQUENCE {
        joinBaseObject      [0]     Name,
        domainLocalID       [1]     DomainLocalID OPTIONAL,
        joinSubset          [2]     ENUMERATED {
        baseObject(O), oneLevel(1), wholeSubtree(2)} DEFAULT baseObject,
        joinFilter          [3]     Filter OPTIONAL,
        joinAttributes      [4]     SEQUENCE SIZE (1 ..MAX) OF JoinAttPair OPTIONAL,
        joinSelection       [5]     EntryInformationSel0ection }

DomainLocalID ::= DirectoryString { ub-domainLocalID }
```

Параметр **DomainLocalID** – это цепочка (строка), которая на местном уровне однозначно идентифицирует удаленный домен, содержащий часть другого DIT.

ПРИМЕЧАНИЕ. – Эта цепочка определяется на местном уровне и не нуждается в регистрации каким-либо регистрирующим органом.

```
JoinAttPair  ::=  SEQUENCE {
        baseAtt                 AttributeType,
        joinAtt                 AttributeType,
        joinContext             SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL}

JoinContextType  ::=  CONTEXT.&id({SupportedContexts} )

SearchResult  ::=  OPTIONALLY-PROTECTED {
        CHOICE {
                searchInfo                  SET {
                    name                                Name OPTIONAL,
                    entries                 [0]         SET OF EntryInformation,
                    partialOutcomeQualifier [2]         PartialOutcomeQualifier OPTIONAL,
                    altMatching             [3]         BOOLEAN DEFAULT FALSE,
                    COMPONENTS OF                       CommonResults },
                uncorrelatedSearchInfo      [0]         SET OF SearchResult } }
```

10.2.2 Аргументы поиска

Аргумент **baseObject** указывает статью объекта (или, возможно, корень), по отношению к которой должен проводиться первичный поиск. Аргумент **baseObject** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **subset** указывает, что первичный поиск должен применяться:

a) только к **baseObject**;

b) только к непосредственно подчиненным объектам основного объекта (**oneLevel**);

c) к основному объекту и ко всем его подчиненным объектам (**wholeSubtree**).

Если основной объект – это обычная статья, то составные статьи должны подсчитываться как одиночная статья в спецификации **subset**. Если основной объект – это порождающий элемент составной статьи, то опция управления поиском **searchFamily** определяет точный режим поиска. Если основной объект – это порожденный член семейства, то члены семейства должны подсчитываться как отдельные статьи.

Аргумент **filter** используется для исключения из области первичного поиска тех статей, которые не представляют интереса. Информация должна выдаваться только о тех статьях, которые удовлетворяют условиям фильтра (см. п. 7.8). При действии стратегии ослабления, заданной пользователем или правилом поиска, фильтр должен быть сначала выражен требуемыми подстановками из правил сопоставления.

При действии стратегии ослабления, заданной пользователем или правилом поиска, либо тем и другим, выдача объема результатов меньше минимума должна вызвать новую установку фильтра с использованием соответствующего ослабления (см. п. 7.8, а также см. ниже для элемента ослабления **SearchArgument**), проводимого до тех пор, пока не будет найдено достаточно статей или не закончатся ослабления, имеющие определения. Точно так же, выдача объема результатов больше максимума должна вызвать новую установку фильтра с использованием соответствующего ужесточения, проводимого до тех пор, пока не будет найдено достаточное количество (либо несколько меньше) статей или не закончатся ужесточения, имеющие определения.

ПРИМЕЧАНИЕ 1. – Если ослабления не предусмотрены правилом поиска, то пользователь может быть вынужден упростить фильтр и попытаться возобновить запрос, либо, альтернативно, задать определяемое пользователем ослабление.

Компонент **familyGrouping** параметра **CommonArguments** используется для логического объединения статей в семейство до применения фильтра, как описано в пп. 7.3.2 и 7.8.3.

При определении положения основного объекта псевдонимы должны быть переименованы в соответствии с установкой параметра управления службой **dontDereferenceAliases**. Псевдонимы в статьях, подчиненных основному объекту, должны быть переименованы во время поиска в соответствии с установкой параметра **searchAliases**. Если параметр **searchAliases** равен **TRUE**, то псевдонимы должны быть переименованы, а если равен **FALSE**, то псевдонимы не должны быть переименованы. Если параметр **searchAliases** равен **TRUE**, то поиск должен продолжаться в поддереве псевдонима статьи.

Аргумент **selection** указывает, какая информация запрашивается из статей (см. п. 7.6). Однако не следует предполагать, что выдаваемые атрибуты будут теми же, что и в запросе, или будут ограничены им.

ПРИМЕЧАНИЕ 2. – DSA, который координирует распределенные операции для связанных статей (т. е. завершает распознавание имени для аргумента операции Поиск, содержащего **joinArguments**, и нуждается в получении совокупности потенциально связанных статей от внешних источников), должен заменить значение **infoTypes**, определяемое протоколом DAP, на параметр **attributeTypesAndValues** для распределенных операций, а также должен включить атрибуты объединения (т. е. атрибуты в наборе, указанном **JoinAttPairjoinAtt** в **JoinArgumentJoinAttributes**) при выборе атрибутов, которые должны выдаваться при распределенных операциях. Однако статьи и производные статьи, которые выдаются пользователю из координирующего DSA, не будут содержать значений атрибутов в информации, выдаваемой DAP, если значение **infoTypes** было равно **attributeTypesOnly**, и поэтому будет выдаваться **EntryInformation** согласно первоначальному запросу пользователя.

Аргумент **pagedResults** используется для запроса результатов операции с постраничным выводом, как описано в 7.9.

Аргумент **matchedValuesOnly** указывает, что некоторые значения атрибутов не должны включаться в выдаваемую информацию статьи. В частности, если атрибут, который будет выдаваться, имеет несколько значений, причем не все значения этого атрибута внесли вклад в соответствие фильтру поиска в его последней действительной форме (т. е. с учетом ослабления правил сопоставления, а фильтр дал значение TRUE посредством элементов фильтра, отличающихся от **present**, то значения, которые не способствовали этому, не включаются в выдаваемую информацию статьи.

Если аргумент **matchedValuesOnly** определен в аргументе **search**, то к атрибутам, которые будут выдаваться, применяется следующая логика обработки:

a) Если фильтр состоит из одного элемента фильтра, то применяются следующие правила:
   - Если тип элемента фильтра – **present**, то параметр **matchedValuesOnly** не влияет на атрибут в этом элементе фильтра.
   - Если тип элемента фильтра является одним из **equality**, **substrings**, **greaterOrEqual**, **lessOrEqual**, **approximateMatch**, **contextPresent** и **extensibleMatch**, а проверка атрибута не дала TRUE, то параметр **matchedValuesOnly** не оказывает влияния на этот атрибут. Если проверка дала TRUE, то значения этого атрибута, который не соответствовал элементу фильтра, исключаются из выдаваемой информации статьи.
   - Если элемент фильтра инвертирован, то аргумент **matchedValuesOnly** не влияет на этот атрибут.

b) Если фильтр сложный (содержит более одного элемента фильтра), то применяются следующие правила:
   - Если фильтр содержит инвертированный (т. е. **not**) фильтр, то аргумент **matchedValuesOnly** не влияет на любой атрибут в рамках инвертированного фильтра.

     ПРИМЕЧАНИЕ 3. – Это применяется также к вложенным инвертированным фильтрам.
   - Аргумент **matchedValuesOnly** не влияет на атрибуты любых элементов фильтров **or** (или), которые оцениваются как FALSE или UNDEFINED.
   - Для атрибута, который попадает в фильтр несколько раз, достаточно одного попадания для оценки TRUE, как описано выше во втором абзаце пункта a), чтобы аргумент **matchedValuesOnly** стал действительным, т. е. один случай действительности отменяет один или большее число случаев игнорирования.
   - Каждый фильтр в фильтре **or** должен быть оценен для аргумента **matchedValuesOnly**, даже если подтверждение фильтра может быть определено прежде, чем закончена полная оценка.

Аргумент **extendedFilter** используется в оборудовании со смешанной версией, чтобы определить фильтр, альтернативный к описанному выше. Когда этот аргумент присутствует, аргумент **filter** (если он присутствует) игнорируется системами второго и последующих изданий. Аргумент **extendedFilter** всегда игнорируется системами первого издания. Ослабление при поиске применяется так же, как для аргумента **filter**.

ПРИМЕЧАНИЕ 4. – Применив оба фильтра, DUA может определить один фильтр, который будет использоваться системами первого издания, и другой фильтр, который будет использоваться в системах второго и последующего изданий при в распределенной обработке запроса Поиск. Системы первого издания не поддерживают полиморфизм атрибута, а также проверки с помощью правил сопоставления.

Аргумент **checkOverspecified** используется для запроса к Справочнику о выдаче элемента **overspecFilter** в **partialOutcomeQualifier** в случае, когда результат операции поиска пуст, а Справочник способен определить, что это является следствием узко заданного фильтра.

Компонент **relaxation** может использоваться, чтобы задать определяемый пользователем параметр **RelaxationPolicy** при помощи конструкции, описанной в п. 16.10 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Подстановки, указанные запросом **search**, не должны выполняться в пределах зависящей от службы административной области, если подстановка приводит поиск в недействительное положение согласно управляющему правилу поиска. Управляющее правило поиска будет нарушено, когда подстановочное правило сопоставления:

a) фактически удаляет один или несколько элементов фильтра из фильтра **search**; или

b) нарушает спецификацию **matchingUse** для типа атрибута (см. п. 16.10.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2).

ПРИМЕЧАНИЕ 5. – Правило сопоставления **nullMatch** обладает свойством удалять из фильтра один или несколько элементов фильтра. При использовании этого правила сопоставления управляющее правило поиска может быть нарушено.

Если операция Поиск выполнена вне зависящей от службы административной области или если управляющее правило поиска не предоставляет компонент **RelaxationPolicy**, то применяется определенный пользователем компонент **RelaxationPolicy**, как описано в п. 16.10.7 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если имеется также **RelaxationPolicy**, определенный правилом поиска, то применяются комбинации согласно следующей процедуре:

1) Определенная правилом поиска базовая стратегия подстановки, если таковая имеется, уже применяется в процессе проверки допустимости поиска. Возможные базовые подстановки, указанные управляющим правилом поиска, применяются, таким образом, *априорно*.

2) Далее должны применяться базовые подстановки и отображение на базе отображения, указанные в запросе **search**, если они имеются. Однако базовые подстановки, которые приводят к нарушению управляющего правила поиска, не должны применяться, а должны игнорироваться. Значение **oldMatchingRule** (если предоставляется) в этом случае применяется к базовому правилу сопоставления, т. е. к правилу, которое применялось бы при отсутствии базовой стратегии подстановки, определяемой правилом поиска.

3) Затем применяются подстановки ослабления/ужесточения, если они имеются согласно запросу **search**, вместе с каким-либо указанным сопоставлением на базе отображения, следуя правилам, определенным в п. 16.10.7 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если подстановка в правило сопоставления сталкивается с пунктом, который вызвал несоответствие с управляющим правилом поиска, то эта конкретная подстановка отменяется полностью, вместе с любыми дальнейшими подстановками, которые, возможно, были определены запросом **search** для этого типа атрибута. Если в течение процесса, определенная в запросе **search** спецификация **minimum** или **maximum** достигнута, то процесс останавливается.

4) Применяются подстановки ослабления или ужесточения, указанные управляющим правилом поиска, за исключением того, что не должно быть подстановки для типов атрибута, для которых были выполнены подстановки ослабления или ужесточения. Таким образом, дальнейшие подстановки ослабления или ужесточения применяются только к правилам сопоставления для типов атрибута, которые пока не были подвергнуты подстановке ослабления или ужесточения. В этой части процесса еще применяются спецификации **minimum** или **maximum** из запроса **search**, а не те, которые определены в указанном управляющем правиле поиска.

Если подстановка, указанная в запросе **search**, предлагает нереализованное правило сопоставления, то остается существующее правило сопоставления. Если эта стратегия будет не в состоянии обеспечить реализованное правило сопоставления, то элемент фильтра дает UNDEFINED.

Пользователь может предлагать, чтобы система обеспечивала ослабление или ужесточение, указав фиктивное правило сопоставления **systemProposedMatch**.

Компонент **extendedArea** указывает уровень ослабления (если больше нуля) или уровень ужесточения (если меньше нуля). Если этот компонент присутствует, то он действует на ослабление или ужесточение, как описано в п. 16.10.7 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Параметр управления поиском **hierarchySelection** определяет посредством цепочки битов иерархический выбор, который будет выполнен в пределах иерархической группы для каждой согласованной (показавшей соответствие) статьи. Он игнорируется для согласованных статей, которые не являются частью какой-либо иерархической группы. Если соответствие показали несколько статей в пределах иерархии, то иерархический выбор не будет приводить к одной и той же статье для выдачи более одного раза. Если этот параметр управления поиском отсутствует, то иерархический выбор не выполняется. Когда он присутствует, возможны следующие варианты по одному или в комбинациях:

a) Значение **self** указывает, что информация статьи должна быть выдана из согласованных статей. Если оно является единственным вариантом, то оно означает отсутствие какого-либо иерархического выбора.

b) Значение **children** указывает, что для каждой согласованной статьи выдается информация статьи из всех иерархических непосредственно порожденных статей, если таковые имеются, от согласованной статьи. Если оно является единственной установкой, то никакая информация не выдается из согласованной статьи.

c) Значение **parent** указывает, что для каждой согласованной статьи выдается информация статьи из иерархической непосредственно родительской статьи, если таковая имеется, для согласованной статьи. Если оно является единственной установкой, то никакая информация не выдается из согласованной статьи.

d) Значение **hierarchy** указывает, что для каждой согласованной статьи выдается информация статьи из всех иерархических родителей. Если оно является единственной установкой, то никакая информация не выдается из согласованной статьи.

e) Значение **top** указывает, что для каждой согласованной статьи выдается информация статьи от вершины иерархии. Никакая информация не выдается из согласованной статьи, если это – единственная установка, за исключением случая, когда согласованная статья является вершиной иерархии.

f) Значение **subtree** указывает, что для каждой согласованной статьи выдается информация статьи из всех ее иерархических порожденных статей, если они имеются. Если оно является единственной установкой, то никакая информация не выдается из согласованной статьи.

g) Значение **siblings** указывает, что для каждой согласованной статьи выдается информация статьи из всех иерархических одноуровневых статей. Никакая информация не выдается из согласованной статьи, если это – единственная установка.

h) Значение **siblingChildren** указывает, что для каждой согласованной статьи выдается информация статьи из иерархических непосредственно порожденных статей всех иерархических одноуровневых статей. Никакая информация не выдается из согласованной статьи и ее одноуровневых статей, если это – единственная установка.

i) Значение **siblingSubtree** указывает, что для каждой согласованной статьи выдается информация статьи из всех порожденных статей всех иерархических одноуровневых статей. Никакая информация не выдается из согласованной статьи и ее одноуровневых статей, если это – единственная установка.

j) Значение **all** указывает, что для каждой согласованной статьи выдается информация статьи из всех статей иерархической группы.

Компонент **searchControlOptions** содержит только опции параметров управления, применимые для операции Поиск. Этот компонент имеет указатели с той же семантикой, что и булевы типы компонентов аргумента поиска. Реализация, обеспечивающая расширение административной службы, должна поддерживать этот компонент. Передающая поддерживающая реализация (например, DUA) в дополнение к установке компонентов булевого типа должна также устанавливать соответствующие биты в этом компоненте (если не применяются значения "по умолчанию"). Если поддерживающая реализация DSA получает запрос **search** с этим компонентом, то она должна игнорировать компоненты булевого типа в этом запросе. Если этот компонент отсутствует в запросе, то установка "по умолчанию" должна распознаваться как сброс всех битов, за исключением нижеследующих случаев:

a) Опция управления поиском **searchAliases** является заменой для компонента аргумента поиска **searchAliases**. Если данный бит установлен, то это соответствует установке компонента **searchAliases** в **TRUE**. Если компонент **searchControlOptions** отсутствует, то значение "по умолчанию" будет соответствовать компоненту **searchAliases**, т. е. если компонент **searchAliases** отсутствует или установлен в **TRUE**, то этот бит устанавливается "по умолчанию".

b) Опция управления поиском **matchedValuesOnly** является заменой для компонента аргумента поиска **matchedValuesOnly**. Если данный бит установлен, то это соответствует установке компонента **matchedValuesOnly** в **TRUE**. Если компонент **searchControlOptions** отсутствует, то значение "по умолчанию" будет соответствовать компоненту **matchedValuesOnly**, т. е. если компонент **matchedValuesOnly** установлен в **TRUE**, то этот бит устанавливается "по умолчанию"; в противном случае он сбрасывается "по умолчанию".

c) Опция управления поиском **checkOverspecified** является заменой для компонента аргумента поиска **checkOverspecified**. Если данный бит установлен, то это соответствует установке компонента **checkOverspecified** в **TRUE**. Если компонент **searchControlOptions** отсутствует, то значение "по умолчанию" будет соответствовать компоненту **checkOverspecified**, т. е. если компонент **checkOverspecified** установлен в **TRUE**, то этот бит устанавливается "по умолчанию"; в противном случае он сбрасывается "по умолчанию".

d) Опция управления поиска **performExactly** указывает, что операция должна выполняться точно согласно уместным правилам сопоставления, как определено или подразумевается в фильтре после подстановки базового правила сопоставления, если оно применимо. Если элемент фильтра **extensibleMatch** обнаруживает нереализованное правило сопоставления, то запрос **search** должен быть отклонен, когда установлена эта опция управления поиском. В противном случае элемент фильтра выдает UNDEFINED. Если операция Поиск начинает свою начальную фазу оценки в пределах зависящей от службы административной области, а ограничение на сопоставление в правиле поиска нарушено, то правило поиска отклоняет подтверждение результата поиска тогда и только тогда, когда установлена эта опция управления поиском.

e) Опция управления поиском **includeAllAreas** применима только в случае, когда компонент **extendedArea** включен со значением, большим или равным нулю. Во всех других случаях она игнорируется. Если ее значение равно **TRUE**, то выполняется включающее ослабление; в противном случае выполняется исключающее ослабление, если оно возможно (см. 13.6 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2).

f) Опция управления поиском **noSystemRelaxation** используется, когда пользователь требует, чтобы стратегии ослабления в DSA не применялись. DSA должен применять базовую стратегию, если отсутствует указанная пользователем базовая стратегия, которая ее отменяет, причем никакие последующие ослабления или ужесточения не должны применяться. Таким образом, этот фильтр никогда не применяется более одного раза для набора статей-кандидатов, если не устанавливаются ослабления от пользователя.

g) Опция управления поиском **dnAttribute** используется для указания, что атрибуты Выделенного имени статьи используются в дополнение к атрибутам статьи при применении фильтра к этой статье. Если она установлена, то она отменяет любые возможные спецификации **dnAttribute** в элементах фильтра **extensibleMatch**. Это применимо также ко всем типам элементов фильтра.

h) Опция управления поиском **matchOnResidualName** применима только в случае, когда установлена опция управления службой **partialNameResolution**. Она используется для указания, что если Справочник способен распознать только часть предполагаемого имени в операции **search**, то проверки AVA нераспознанных RDN должны интерпретироваться как элементы фильтра **equality**, соединенные логическим И. Эти элементы фильтра соединены логическим И с фильтром поиска как для поисковой оценки по правилам поиска, так и для сопоставления статьи.

i) Опция управления поиском **entryCount** указывает, что подсчет статей должен присутствовать в результате операции **search** в том случае, когда был превышен лимит размера от управления службой или административный лимит размера. Опция **entryCount** показывает, сколько статей было бы выдано, если бы лимит размера не учитывался. Данная опция управления поиском игнорируется, если установлена опция управления службой **subentries**.

j) Опция управления поиском **useSubset** указывает, что компонент **imposedSubset** правила поиска должен игнорироваться (см. п. 16.10.9 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2).

k) Опция управления поиском **separateFamilyMembers** указывает, что члены семейства выдаются в виде отдельных статей, а не включены в выделенный атрибут **family-information**.

l) Опция управления поиском **searchFamily** определяет, как выполняется поиск, если базовый объект является порождающим объектом для составного атрибута. Эта опция игнорируется, если базовый объект не является порождающим, либо если компонент **entryOnly** установлен в **CommonArguments** или **ChainingArguments**. Если эта опция установлена, то операция выполняется только для составной статьи, а каждый член семейства считается отдельной статьей для спецификаций **subset** и **sizeLimit**. Если опция **searchFamily** не установлена, то составная статья рассматривается как одна статья для спецификации **subset**.

ПРИМЕЧАНИЕ 6. – Последнее утверждение означает, например, что если **subset** установлен в **baseObject**, а **familyGrouping** установлен в **entryOnly**, то каждый отдельный член семейства является предметом поиска.

Аргумент **joinArguments** используется, чтобы определить дополнительные части Справочника, в которых нужно проводить поиск для идентификации статей и доступа к статьям, связанным со статьями первичного поиска, а также для определения атрибутов, которые нужно использовать при объединении связанных статей. Хотя аргумент определен как SEQUENCE, порядок следования аргументов **joinArgument** не существенен.

ПРИМЕЧАНИЕ 7. – Когда **joinArguments** указан, считается, что первичный поиск и каждый дополнительный поиск дают набор промежуточных результатов. Каждый набор промежуточных результатов, следующих из спецификации **joinArgument**, будет объединен с результатами первичного поиска, а все объединения будут выполнены перед выдачей какого-либо результата в **SearchResult**. Промежуточные результаты не видны пользователям Справочника.

Аргумент **joinBaseObject** указывает статью объекта (или, возможно, корень), для которой проводится каждый дополнительный поиск. Аргумент **joinBaseObject** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **domainLocalID** факультативно определяет отдельный DIT, в котором должен быть начат поиск **joinBaseObject**. При отсутствии этого аргумента поиск **joinBaseObject** должен быть начат во всех DIT, известных DSA.

Аргумент **joinSubset** указывает, что дополнительный поиск должен применяться:

a) только к j**oinBaseObject**;

b) только к непосредственно подчиненным объектам связанного базового объекта (**oneLevel**);

c) к связанному базовому объекту и ко всем его подчиненным объектам (**wholeSubtree**).

Аргумент **joinFilter** используется для устранения статей, которые не представляют интереса, из дополнительной области поиска. Только информация, которая удовлетворяет **joinFilter**, будет рассматриваться для соединения со связанными статьями. Если **joinFilter** не указан, то будет использоваться значение из компонента **filter** параметра **SearchArgument**. Если компонент **filter** в параметре **SearchArgument** отсутствует, то для этого компонента будет использоваться значение "по умолчанию". Если **joinFilter** присутствует, то он будет обработан согласно правилам для **extendedFilter**.

Аргумент **joinAttributes** используется для определения пар атрибутов, которые следует использовать при соединении статей из первичного поиска со статьями из дополнительного поиска. Статья из первичного поиска ("первичная статья") считается связанной со статьей из дополнительного поиска ("дополнительная статья"), если имеется компонент **joinAttrPair** и что выполняются следующие условия:

a) Первичная статья имеет значение для типа атрибута, указанного параметром **baseAtt**.

b) Дополнительная статья имеет значение для типа атрибута, указанного параметром **joinAtt**.

c) Одно из значений атрибута в первичной статье и одно из значений атрибута в дополнительной статье равны согласно следующим правилам:

   i) Если типы атрибута одинаковы, то применяется правило сопоставления на предмет эквивалентности для этого типа атрибута.

   ii) Если типы атрибута не одинаковы, но имеют одинаковый синтаксис, то применяется правило сопоставления на предмет эквивалентности для типа атрибута, указанного для первичной статьи.

   iii) Если присутствует компонент **joinContexts**, то только значения атрибутов указанных контекстов могут использоваться при оценке согласно вышеуказанному правилу i) или ii). Если **joinContexts** отсутствует, то при оценке согласно вышеуказанному правилу i) или ii) могут использоваться значения атрибутов всех контекстов.

При оценке **joinAttributes** для потенциальных соединений подтипы соединяемых атрибутов должны игнорироваться. Только явно указанные аргументы **baseAtt** и **joinAtt** будут использоваться для оценки потенциального соединения.

Если применяется правило эквивалентности, а оценка дает FALSE или UNDEFINED, то статьи не рассматриваются как связанные.

Если нет подходящего правила сопоставления, которое может применяться при вышеуказанном условии c), то статьи не рассматриваются как связанные.

ПРИМЕЧАНИЕ 8. – Должны быть приняты меры предосторожности, чтобы предотвратить неумышленный поиск бесцельных данных при определении соединений, охватывающих атрибуты с несколькими значениями. Например, если статья использует атрибут с несколькими значениями, вроде идентификатора служащего для обозначения членства в комитете, то описание этого атрибута с несколькими значениями при выполнении соединения может привести к выдаче несвязанного набора, содержащего имена членов группы, номера телефонов, адреса электронной почты и т. п. Если, однако, внешние соединения определены, то все найденные статьи будут выдаваться, даже если они не связаны.

Аргумент **joinSelection** используется для устранения атрибутов, которые не представляют интереса, из промежуточного результата дополнительного поиска.

Аргумент **joinType** используется, чтобы указать тип соединения, которое будет выполняться для связанных статей, следующим образом:

a) Если указано **innerJoin**, то образованный набор статей будет содержать только те статьи, для которых соединение выполнено на основе пар атрибутов, указанных в **joinAttributes**. Каждая образованная статья будет содержать все соответствующие связанные статьи в виде значений атрибута **relatedEntry**.

b) Если указано **leftOuterJoin**, то образованный набор статей будет содержать все статьи, выбранные при первичном поиске; все статьи, для которых соединение было выполнено на основе пар атрибутов, указанных в **joinAttributes**, будут содержать все соответствующие связанные статьи в виде значений атрибута **relatedEntry**.

c) Если указано **fullOuterJoin**, то образованный набор статей будет содержать все статьи из первичного и дополнительного поисков; все статьи, для которых соединение было выполнено на основе пар атрибутов, указанных в **joinAttributes**, будут содержать все соответствующие связанные статьи в виде значений атрибута **relatedEntry**, а не в виде явных статей.

Попытка соединения не выполняется, если значение **joinAttributes** не содержит, по крайней мере, один параметр **JoinAttPair** и если не каждый **JoinAttPair** действителен для правил сопоставления. В этом случае никакая попытка соединения не выполняется, и формируется следующий объединенный вывод для каждого **JoinAttPair**, в зависимости от типа соединения:

Тип соединения	Объединенный вывод
inner-join	пустой
left-outer-join	только результаты первичного поиска
full-outer-join	результаты первичного и связанного поиска

По-другому, статьи могут соединяться только тогда, когда они могут обеспечить все соответствующие значения соединенного атрибута.

Результаты соединения должны содержать все комбинации сопоставленных соединенных атрибутов.

ПРИМЕЧАНИЕ 9. – Например, рассмотрим A, B и C как статьи из первичного поиска, а P, Q, R как статьи из дополнительного поиска с использованием J – соответствующего значения **JoinAttPair**, а затем предположим, что следующие соответствия получились в результате J:

– A c P, A c Q, A c R

– B c Q

– C c P и C c Q

Тогда соединенные результаты будут содержать:

– A c {P,Q,R}

– B c {Q}

– C c {P,Q}

даже если результаты Q встречаются три раза.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

10.2.3 Результаты поиска

Запрос завершается успешно согласно параметрам управления доступом, если **baseObject** найден, независимо от того, имеются ли подчиненные объекты для выдачи в ответе, и если нет ограничений на службу в пределах зависящей от службы административной области, которые препятствуют выполнению операции Поиск.

ПРИМЕЧАНИЕ 1. – Как следствие этого, результат нефильтрованного поиска, примененного к одной статье, может быть не идентичным операции чтения, которая пытается запросить тот же самый набор атрибутов статьи. Это происходит из-за того, что последняя операция должна выдавать **AttributeError**, если ни один из выбранных атрибутов не существует в статье.

Параметр **name –** это выделенное имя статьи или имя псевдонима статьи, описанные в п. 7.7. Он присутствует только в случае, когда псевдоним был переименован, RDN были приведены к первичным RDN, либо был применен контекстный выбор, а имя, которое должно выдаваться, отличается от имени **baseObject**, указанного в аргументе операции.

Параметр **entries** переносит запрошенную информацию из каждой статьи (в количестве нуль или больше), которая удовлетворяет фильтру (см. п. 7.5). На имена, входящие в **entries**, могут влиять контексты, как описано для **Name** в п. 7.7. Информация статьи может содержать информацию семейства, как требуется элементом **familyReturn** в **EntryInformationSelection**. Взаимодействие между **familyGrouping** и **familyReturn** определяется при четырехфазовой операции фильтра и при последующей оценки того, что должно выдаваться, как описано в п. 7.8.3.

Параметр **partialOutcomeQualifier** такой же, как описанный в п. 10.1.3.

ПРИМЕЧАНИЕ 2. – Если выдаваемая информация статьи неполна для конкретной статьи, то на это указывает параметр **incompleteEntry** в выдаваемой информации статьи.

Параметр **altMatching** указывает, что правило сопоставления не применялось, как и было определено в запросе **search**.

Атрибут **appliedRelaxation** в элементе **notifications** параметра **CommonResults** должен использоваться для формирования списка атрибутов фильтра, которые подвергались ослаблению или ужесточению, которое отличается от проведенных элементом **basic** из стратегии ослабления (см. п. 5.12.16 Рекомендации МСЭ-Т X.520 | ИСО/МЭК 9594-6).

Параметр **uncorrelatedSearchInfo** такой же, как описанный в п. 10.1.3 параметр **uncorrelatedListInfo**.

Информация **CommonResults** (см. п. 7.4) содержит параметры безопасности, применяемые к ответу. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) включается в результаты.

10.2.4 Администрирование службы

Административный орган может устанавливать зависящие от службы административные области, как определено в разделе 7 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Это позволяет административному органу управлять службой путем ограничения операции Поиск в отношении областей DIT, в которых может проводиться поиск типа поиска, который может быть сформирован, вида информации, которая может выдаваться, и т. д., и путем определения правил поиска.

10.2.5 Ошибки поиска

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

Если поиски выполнены в пределах зависящих от службы административных областей, то может быть выдан ряд дополнительных весьма детальных сведений об ошибке, как подробно описывается в разделе 13.

10.2.6 Точки принятия решения операции Поиск при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если **basic-access-control** влияет на часть DIT, в которой выполняется поиск, то применяется следующая последовательность управления доступом:

1) Для статьи, указанной аргументом **baseObject**, не требуется никакого специального разрешения.

   ПРИМЕЧАНИЕ 1. – Если **baseObject** находится в пределах **SearchArgument** (т. е. когда аргумент **subset** имеет значение **baseObject** или **wholeSubtree**), то применяются параметры управления доступом, указанные в пунктах от 2) до 5).

2) Для каждой статьи в пределах **SearchArgument**, являющейся кандидатом для рассмотрения, требуется разрешение *Browse*. Статьи, для которых это разрешение не дано, игнорируются.

3) Аргумент **filter** применяется к каждой статье, оставленной на рассмотрение после выполнения пункта 2), согласно следующей процедуре:

   a) Для каждого **FilterItem**, определяющего атрибут, требуется разрешение *FilterMatch* (*Соответствие фильтру*) для этого типа атрибута до того, как **FilterItem** сможет дать оценку TRUE или FALSE. **FilterItem**, для которого этого разрешения нет, дает оценку UNDEFINED.

   b) Для каждого **FilterItem**, который дополнительно определяет значение атрибута, требуется разрешение *FilterMatch* для каждого записанного значения атрибута, которое будет рассматриваться с целью сопоставления. Если имеется значение, которое соответствует **FilterItem** и для которого дано разрешение, то **FilterItem** дает оценку TRUE, в противном случае – оценку FALSE.

4) Если присутствует аргумент **joinCriteria**, то он применяется к каждой статье, оставленной на рассмотрение после выполнения пункта 3), согласно следующей процедуре:

   a) Для каждого **JoinCriterialtem**, определяющего атрибут, требуется разрешение *FilterMatch* для этого типа атрибута до того, как **JoinCriterialtem** сможет дать оценку TRUE или FALSE. **JoinCriterialtern**, для которого этого разрешения нет, дает оценку UNDEFINED.

   b) Для каждого **JoinCriterialtem**, который дополнительно определяет значение атрибута, требуется разрешение *FilterMatch* для каждого записанного значения атрибута, которое будет рассматриваться с целью сопоставления. Если имеется значение, которое соответствует **JoinCriterialtem** и для которого дано разрешение, то **JoinCriterialtem** дает оценку TRUE, в противном случае – оценку FALSE.

5) После применения процедур, определенных в пунктах 2)–4), статья либо выбирается, либо сбрасывается. Если вследствие применения этих параметров управления для всего просматриваемого поддерева не было выбрано ни одной статьи (за исключением любых **ContinuationReferences** в **partialOutcomeQualifier**), либо если разрешение *DiscloseOnError* не предоставлено статье, указанной аргументом **baseObject**, то операция оканчивается неудачей и выдается уведомление **nameError** с указанием причины **noSuchObject**. Элемент **matched** будет содержать имя следующей старшей статьи, для которой предоставлено разрешение *DiscloseOnError*, или имя корня DIT (т. е. пустую последовательность **RDNSequence**). В остальных случаях операция оканчивается успешно, но никакая информация подчиненной статьи не выдается.

   ПРИМЕЧАНИЕ 2. – В случае выдаваемого параметра **nameError** пустая последовательность **RDNSequence** может использоваться в DSA, который не имеет доступа ко всем старшим статьям.

   ПРИМЕЧАНИЕ 3. – Стратегия безопасности может предотвращать раскрытие информации типа знаний, которая иначе была бы передана как **ContinuationReferences** в **partialOutcomeQualifier**. Если такая стратегия действует, и если DUA ограничивает службу, указывая **chainingProhibited**, то Справочник может выдать уведомление **serviceError** с указанием причины **chainingRequired**. В противном случае **ContinuationReference** опускается из **partialOutcomeQualifier**.

6) В остальных случаях для каждой выбранной статьи выдается информация следующим образом:

   a) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться только типы атрибутов, то для каждого типа атрибута, который должен выдаваться, требуется разрешение *Read*. Если разрешение не дано, то такой тип атрибута опускается из параметра **EntryInformation**. Если вследствие применения этих параметров управления никакая информация о типе атрибута не выбрана, то выдается элемент **EntryInformation**, но никакая информация о типе атрибута не переносится в нем (т. е. элемент **SET OF CHOICE** опущен или пустой).

b) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться типы и значения атрибутов, то для каждого типа атрибута и для каждого значения, которые должны выдаваться, требуется разрешение *Read*. Если разрешение для типа атрибута не дано, то такой атрибут опускается из **EntryInformation**. Если разрешение для значения атрибута не дано, то это значение опускается из соответствующего атрибута. Если разрешение не дано для любого из значений атрибута, то выдается элемент **Attribute,** содержащий пустое множество **SET OF AttributeValue**. Если вследствие применения этих параметров управления никакая информация атрибутов не выбрана, то выдается элемент **EntryInformation**, но никакая информация атрибутов не переносится в нем (т. е. элемент **SET OF CHOICE** опущен или пустой).

ПРИМЕЧАНИЕ 4. – Если разрешение *DiscloseOnError* не предоставлено для статьи, указанной аргументом **baseObject**, то параметр **partialOutcomeQualifier** с указанием **limitProblem** или **unavailableCriticalExtensions** не должен выдаваться, поскольку он может скомпрометировать безопасность этой статьи.

10.2.6.1 Точки принятия решения операции Поиск при управлении базовым доступом с дополнительными операциями поиска

Если присутствует аргумент **joinArguments** и если **basic-access-control** влияет на часть DIT, в которой выполняется поиск, то применяется следующая последовательность управления доступом для каждого дополнительного поиска:

1) Для статьи, указанной аргументом **joinBaseObject**, не требуется никакого специального разрешения.

ПРИМЕЧАНИЕ 1. – Если **joinBaseObject** находится в пределах **joinArgument** (т. е. когда аргумент **joinSubset** имеет значение **baseObject** или **wholeSubtree**), то применяются параметры управления доступом, указанные в пунктах 2)–6).

2) Для каждой статьи в пределах **joinArgument**, которая является кандидатом для рассмотрения, требуется разрешение *Browse*. Статьи, для которых это разрешение не дано, игнорируются.

3) Если аргумент **joinFilter** присутствует, то он применяется для каждой статьи, оставленной на рассмотрение после выполнения пункта 2), согласно следующей процедуре:

a) Для каждого **FilterItem**, который определяет атрибут, требуется разрешение *FilterMatch* для этого типа атрибута до того, как **FilterItem** сможет дать оценку TRUE или FALSE. **FilterItem**, для которого этого разрешения нет, дает оценку UNDEFINED.

b) Для каждого **FilterItem**, который дополнительно определяет значение атрибута, требуется разрешение *FilterMatch* для каждого записанного значения атрибута, которое будет рассматриваться с целью сопоставления. Если имеется значение, которое соответствует **FilterItem** и для которого дано разрешение, то **Filteritem** дает оценку TRUE, в противном случае – оценку FALSE.

4) Если аргумент **joinFilter** отсутствует, то аргумент **filter** применяется к каждой статье, оставленной на рассмотрение после выполнения пункта 2), согласно следующей процедуре:

a) Для каждого **FilterItem**, который определяет атрибут, требуется разрешение *FilterMatch* для этого типа атрибута до того, как **FilterItem** сможет дать оценку TRUE или FALSE. **FilterItem**, для которого этого разрешения нет, дает оценку UNDEFINED.

b) Для каждого **FilterItem**, который дополнительно определяет значение атрибута, требуется разрешение *FilterMatch* для каждого записанного значения атрибута, которое будет рассматриваться с целью сопоставления. Если имеется значение, которое соответствует **FilterItem** и для которого дано разрешение, то **Filteritem** дает оценку TRUE, в противном случае – оценку FALSE.

5) После применения процедур, определенных в пунктах 2)–4), статья либо выбирается, либо сбрасывается. Если вследствие применения этих параметров управления для всего просматриваемого поддерева не было выбрано ни одной статьи (за исключением любых **ContinuationReferences** в **partialOutcomeQualifier**), либо если разрешение *DiscloseOnError* не предоставлено статье, указанной аргументом **baseObject**, то операция оканчивается неудачей и выдается уведомлением **nameError** с указанием причины **noSuchObject**. Элемент **matched** будет содержать имя следующей старшей статьи, для которой предоставлено разрешение *DiscloseOnError*, или имя корня DIT (т. е. пустую последовательность **RDNSequence**). В остальных случаях операция заканчивается успешно, но никакая информация подчиненной статьи не переносится в нем.

ПРИМЕЧАНИЕ 2. – В случае выдаваемого параметра **nameError** пустая последовательность **RDNSequence** может использоваться в DSA, который не имеет доступа ко всем старшим статьям.

ПРИМЕЧАНИЕ 3. – Стратегия безопасности может предотвращать раскрытие информации в виде знаний, которая иначе была бы передана как **ContinuationReferences** в **partialOutcomeQualifier**. Если такая стратегия действует, и если DUA ограничивает службу, указывая **chainingProhibited**, то Справочник может выдать уведомление **serviceError** с указанием причины **chainingRequired**. В противном случае **ContinuationReference** опускается из **partialOutcomeQualifier**.

6) В остальных случаях для каждой выбранной статьи выдается информация следующим образом:

   a) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться только типы атрибутов, то для каждого типа атрибута, который должен выдаваться, требуется разрешение *Read*. Если разрешение не дано, то такой тип атрибута опускается из параметра **EntryInformation**. Если вследствие применения этих параметров управления никакая информация о типе атрибута не выбрана, то выдается элемент **EntryInformation**, но никакая информация о типе атрибута не переносится в нем (т. е. элемент **SET OF CHOICE** опущен или пустой).

   b) Если элемент **infoTypes** в параметре **selection** указывает, что должны выдаваться типы и значения атрибутов, то для каждого типа атрибута и для каждого значения, которые должны выдаваться, требуется разрешение *Read*. Если разрешение для типа атрибута не дано, то такой атрибут опускается из **EntryInformation**. Если разрешение для значения атрибута не дано, то это значение опускается из соответствующего атрибута. Если разрешение не дано для любого из значений атрибута, то выдается элемент **Attribute**, содержащий пустое множество **SET OF AttributeValue**. Если вследствие применения этих параметров управления никакая информация атрибутов не выбрана, то выдается элемент **EntryInformation**, но никакая информация атрибутов не переносится в нем (т. е. элемент **SET OF CHOICE** опущен или пустой).

   ПРИМЕЧАНИЕ 4. – Если разрешение *DiscloseOnError* не предоставлено для статьи, указанной аргументом **baseObject**, то параметр **partialOutcomeQualifier** с указанием **limitProblem** или **unavailableCriticalExtensions** не должен выдаваться, поскольку он может скомпрометировать безопасность этой статьи.

10.2.6.2 Переименование псевдонима во время поиска

Для переименования псевдонима в процессе операции **search** (выполняемого путем установки параметра **searchAliases** в **TRUE**) не требуется никаких специальных разрешений. Однако если для каждой встречающейся статьи переименование псевдонима приводит к параметру **ContinuationReference**, который выдается в **partialOutcomeQualifier**, то применяется следующая последовательность параметров управлений доступом: разрешение *Read* требуется для статьи псевдонима, для атрибута **aliasedEntryName** и для одного значения, которое он содержит. Если любое из этих разрешений не предоставлено, то **ContinuationReference** опускается из **partialOutcomeQualifier**. Эти параметры управления доступом должны применяться также к **continuationReference**, который получен в ответе от другого DSA. Таким образом, DSA будет отслеживать все **continuationReferences**, независимо от того, были они созданы на месте или в удаленном пункте.

ПРИМЕЧАНИЕ. – В дополнение к параметрам управления доступом, описанным выше, стратегия безопасности может предотвращать раскрытие информации в виде знаний, которая иначе была бы передана как **ContinuationReferences** в **partialOutcomeQualifier**. Если такая стратегия действует, и если DUA ограничивает службу, указывая **chainingProhibited**, то Справочник может выдать уведомление **serviceError** с указанием причины **chainingRequired**. В противном случае **ContinuationReference** опускается из **partialOutcomeQualifier**.

10.2.6.3 Нераскрытие неполных результатов

Если в **EntryInformation** возвращается неполный результат, т. е. некоторые из атрибутов или значений атрибута опущены из-за применения соответствующих параметров управления доступом, то элемент **incompleteEntry** должен быть установлен в **TRUE**, если разрешение *DiscloseOnError* предоставлено, по крайней мере, для одного типа атрибута, отсутствующего в результате, или, по крайней мере, для одного значения атрибута, отсутствующего в результате (для типа атрибута, для которого было предоставлено разрешение *Read*).

10.2.7 Точки принятия решения операции Поиск при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на часть DIB, в которой выполняется операция **search**, то применяется следующее управление доступом:

1) Если разрешение на уровне статьи на основе правила не дано для статьи, указанной аргументом **baseObject**, то выдается уведомление **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) При действии процедуры **rule-based-access-control** каждая статья в пределах **SearchArgument**, для которой доступ на уровне статьи отклонен, игнорируется.

3) Применение процедуры **basic-access-control** для статей описано в п. 10.2.6 2).

4) При применении параметра **filter** игнорируются значения атрибута, доступ к которым отклонен процедурой **rule-based-access-control**.

5) Применение процедуры **basic-access-control** для **filter** определено в п. 10.2.6 3) и 4).

6) Для любой выбранной статьи:

   a) для каждого типа атрибута, который может выдаваться согласно **rule-based-access-control**, должен быть предоставлен доступ, по крайней мере, к одному значению атрибута этого типа;

   b) значения атрибута, доступ к которым отклонен согласно **rule-based-access-control**, не должны выдаваться.

7) Процедура **basic-access-control** применяется к выдаваемой информации, как определено в п. 10.2.6 5).

11 Операции модификации Справочника

Имеются четыре операции модификации Справочника: **addEntry** (Добавление статьи), **removeEntry** (Удаление статьи), **modifyEntry** (Модификация статьи) и **modifyDN** (Модификация DN), которые определяются в пп. 11.1–11.4 соответственно.

ПРИМЕЧАНИЕ 1. – При каждой из этих операций указывается целевая статья посредством ее выделенного имени.

ПРИМЕЧАНИЕ 2. – Успешность выполнения операций **addEntry**, **removeEntry** и **modifyDN** может зависеть от физического распределения DIB по Справочнику. При неудаче должно выдаваться уведомление **updateError** с указанием причины **affectsMultipleDSAs**. См. Рекомендацию МСЭ-Т X.518 | ИСО/МЭК 9594-4.

ПРИМЕЧАНИЕ 3. – Результаты операций будут неопределенными в случае отказа нижележащего механизма связи. Пользователь должен применять операции опроса Справочника, чтобы проверить, выполнена ли предпринятая операция модификации или нет.

11.1 Добавление статьи

11.1.1 Синтаксис добавления статьи

Операция **addEntry** используется для добавления статьи-листа к DIT (либо статьи объекта, либо статьи псевдонима). Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Когда затребовано, Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
addEntry OPERATION  ::=  {
        ARGUMENT        AddEntryArgument
        RESULT          AddEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-addEntry }

AddEntryArgument  ::=  OPTIONALLY-PROTECTED {
        SET{
                object                  [0]     Name,
                entry                   [1]     SET OF Attribute,
                targetSystem            [2]     AccessPoint OPTIONAL,
                COMPONENTS OF                   CommonArguments} }

AddEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.1.2 Аргументы добавления статьи

Аргумент **object** идентифицирует статью, которая будет добавлена. Ее непосредственно предшествующая (старшая) статья, которая уже должна существовать для успешного выполнения операции, определяется путем удаления последнего компонента RDN (который принадлежит создаваемой статье). Аргумент **object** может быть альтернативным именем и содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Последний компонент RDN должен быть первичным RDN и должен содержать все выделенные значения с их контекстными списками для всех атрибутов, присутствующих в RDN. Если какой-либо элемент **AttributeTypeAndDistinguishedValue** в последнем компоненте RDN дан без альтернативных выделенных значений, то должно использоваться единственное имеющееся значение в качестве единственного выделенного значения для этого атрибута.

Аргумент **entry** содержит информацию атрибута, которая совместно с информацией RDN образует создаваемую статью. Справочник должен обеспечивать, чтобы статья соответствовала схеме Справочника. Если создаваемая статья – это псевдоним, то проверка того, что атрибут **aliasedEntryName** указывает на действительную статью, не проводится.

Аргумент **targetSystem** указывает на DSA, которое должно принять новую статью. Если этот аргумент отсутствует, то имеется в виду тот DSA, который содержит предшествующий объект для нового объекта. Если аргумент присутствует, то это будет DSA с указанным **AccessPoint**. Если добавляются подстатьи, то этот параметр должен отсутствовать.

Если аргумент присутствует, то бит **targetSystem** параметра **criticalExtensions** в **CommonArguments** должен быть установлен, указывая, что это расширение является критическим.

ПРИМЕЧАНИЕ 1. – Если выбор указанного или подразумеваемого DSA вызывает конфликт с местной административной стратегией, то операция не выполняется и выдается уведомление об ошибке.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Опция **dontDereferenceAlias** игнорируется (и трактуется как установленная), если бит критического расширения **useAliasOnUpdate** не установлен в **criticalExtensions**. Таким образом, псевдонимы переименовываются этой операцией только тогда, когда опция **dontDereferenceAlias** не установлена, а бит **useAliasOnUpdate** установлен. Компонент **sizeLimit** игнорируется, если он присутствует. Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

ПРИМЕЧАНИЕ 2. – Операции обновления, которые охватывают переименование имени псевдонима, будут всегда получать отказ, если они сталкиваются с агентами DSA первого издания.

11.1.3 Результаты добавления статьи

Если запрос завершается успешно, то должен быть выдан некоторый результат. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) параметра **CommonResultsSeq** (см. п. 7.4) включается в результаты. Если результат операции не должен быть подписан Справочником, то никакая информация не должна переноситься в таком результате.

11.1.4 Ошибки добавления статьи

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

11.1.5 Точки принятия решения операции Добавление статьи при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой basic-**access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если basic-access-control влияет на статью, которая добавляется, то применяется следующая последовательность параметров управления доступом:

1) Для статьи, указанной аргументом **object**, не требуется никакого специального разрешения.

   ПРИМЕЧАНИЕ 1. – Стратегия безопасности может не позволять пользователям Справочника добавлять статьи за границами DSA (например, с использованием аргумента **targetSystem**). В этом случае может выдаваться подходящий указатель **nameError**, **serviceError**, **securityError** или **updateError**, при условии что это не компрометирует наличие непосредственно предшествующей статьи. Если это происходит (т. е. разрешение *DiscloseOnError* не предоставлено для предшествующей статьи), то к предшествующей статье должна применяться процедура, определенная в п. 7.11.3.

2) Если статья с выделенным именем, равным аргументу **object**, уже существует, то операция заканчивается неудачей согласно п. 11.1.5.1 a).

3) Для новой добавляемой статьи требуется разрешение *Add (Добавление)*. Если это разрешение не предоставляется, то операция заканчивается неудачей согласно п. 11.1.5.1 b).

   ПРИМЕЧАНИЕ 2. – Разрешение *Add* должно предоставляться в виде "**prescriptiveACI**" при попытке добавления статьи и в виде "**prescriptiveACI**" или "**subentryACI**" при попытке добавления подстатьи.

4) Для каждого типа атрибута и для каждого значения, которые должны быть добавлены, требуется разрешение *Add*. Если какое-либо разрешение отсутствует, то операция заканчивается неудачей согласно подразделу 11.1.5.1 c).

11.1.5.1 Выдача уведомлений об ошибке

Если операция заканчивается неудачей, как определено в п. 11.1.5, то применяется следующая процедура:

a) Если операция заканчивается неудачей согласно п. 11.1.5 2), то действительно одно из выдаваемых уведомлений об ошибке. Если разрешение *DiscloseOnError* или *Add* дано для существующей статьи, то выдается параметр **updateError** с указанием причины **entryAlreadyExists**. В противном случае последует процедура, описанная в п. 7.11.3, применительно к добавляемой статье.

b) Если операция заканчивается неудачей согласно п. 11.1.5 3), то последует процедура, описанная в п. 7.11.3, применительно к добавляемой статье.

c) Если операция заканчивается неудачей согласно п. 11.1.5 4), то действительно выдаваемое уведомление об ошибке **securityError** с указанием причины **insufficientAccessRights** или **noInformation**.

11.1.6 Точки принятия решения операции Добавление статьи при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на часть DIB, в которой выполняется операция **addEntry**, то применяется следующая последовательность управления доступом:

1) Если разрешение на уровне статьи на основе правила не дано для непосредственно предшествующей статьи, то выдается уведомление **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) Процедура **basic-access-control** применяется согласно 11.1.5.

11.2 Удаление статьи

11.2.1 Синтаксис удаления статьи

Операция Удаление статьи используется для удаления из DIT статьи-листа (статьи объекта, члена семейства или статьи псевдонима), либо не являющегося листом порождающего объекта с его порожденными объектами. Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Когда затребовано, Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
removeEntry OPERATION ::= {
        ARGUMENT        RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          { nameError I serviceError I referral I securityError I updateError }
        CODE            id-opcode-removeEntry }

RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]   Name,
                COMPONENTS OF                 CommonArguments } }

RemoveEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.2.2 Аргументы удаления статьи

Аргумент **object** идентифицирует статью, которая будет удалена. Аргумент **object** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Опция **dontDereferenceAlias** игнорируется (и трактуется как установленная), если бит критического расширения **useAliasOnUpdate** не установлен в **criticalExtensions**. Таким образом, псевдонимы переименовываются этой операцией только тогда, когда опция **dontDereferenceAlias** не установлена, а бит **useAliasOnUpdate** установлен. Компонент **sizeLimit** игнорируется, если он присутствует. Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

ПРИМЕЧАНИЕ. – Операции обновления, которые охватывают переименование имени псевдонима, будут всегда получать отказ, если они наталкиваются на DSA первого издания.

Компонент **FamilyGrouping** может быть установлен следующим образом:

– значение **entryOnly** является значением "по умолчанию" для данной операции. Статья, которая будет удалена, должна быть статьей-листом.

– значение **compoundEntry** может быть определено для порождающего объекта. Все члены составной статьи удаляются. Если целевой объект не является порождающим объектом, то операция закончится неудачей и выдаст уведомление **updateError** с указанием причины **notAncestor**. Операция также завершится неуспешно с выдачей соответствующего уведомления об ошибке, если невозможно удалить всех членов, например, из-за аспектов безопасности.

Если компонент **FamilyGrouping** отсутствует или установлен в любое другое значение, отличное от вышеуказанных, то предполагается значение **entryOnly**.

11.2.3 Результаты удаления статьи

Если запрос завершается успешно, то должен быть выдан некоторый результат. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) параметра **CommonResultsSeq** (см. п. 7.4) включается в результаты. Если результат операции не должен быть подписан Справочником, то никакая информация не должна переноситься в таком результате.

Если выбрана информация семейства с помощью параметра **familyReturn** в **EntryInformationSelection**, то выдается информация, определенная в п. 7.6.4.

Информация, выдаваемая в компоненте **information**, соответствует состоянию DIB после (успешной) операции Модификация статьи.

11.2.4 Ошибки удаления статьи

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

11.2.5 Точки принятия решения операции Удаление статьи при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если **basic-access-control** влияет на удаляемую статью, то применяются следующие параметры управлений доступом:

– Для удаляемой статьи требуется разрешение *Remove* (*Удаление*). Если это разрешение не предоставлено, то операция заканчивается неудачей согласно п. 7.11.1.

ПРИМЕЧАНИЕ. – Никаких специальных разрешений не требуется для любого из атрибутов и значений атрибута, имеющихся в удаляемой статье.

11.2.6 Точки принятия решения операции Удаление статьи при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на удаляемую статью, то применяется следующая последовательность управления доступом:

1) Если разрешение на уровне статьи на основе правила не предоставлено для целевой статьи, то операция заканчивается неудачей и выдает **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) Процедура на уровне статьи **basic-access-control** применяется согласно п. 11.2.5.

3) Если доступ на основе правила не предоставлен для какого-либо значения атрибута, то оно не должно быть удалено.

4) Если разрешение для RDN на основе правила не дано, то ни одно из значений атрибута RDN не должно быть удалено. Если все значения атрибута удалены, то этот атрибут удаляется из статьи. Если все атрибуты удалены, то статья удаляется из DIT. Если, по крайней мере, одно значение атрибута удалено, а инициатор не имеет разрешения для RDN, то операция завершается успешно, но статья остается в DIT с одним или большим количеством атрибутов.

   ПРИМЕЧАНИЕ 1. – Если не все значения контекста метки для выделенных значений статьи имеют одинаковое значение, то может не поддерживаться стратегия управления доступом на основе правила.

5) Если при выполнении процедуры **rule-based-access-control** разрешение для RDN дано, но разрешение доступа, по крайней мере, к одному из других значений атрибута не дано, то RDN не удаляется, а операция заканчивается неудачей и выдает **securityError** с указанием причины **insufficientAccessRights**. Вопрос об удалении или не удалении других значений атрибута, для которых инициатор имеет разрешение доступа, является местным вопросом.

   ПРИМЕЧАНИЕ 2. – Это показывает инициатору, что существует, по крайней мере, одно значение атрибута, которое недоступно.

6) Если все атрибуты статьи удалены, то статья удаляется из DIT, а операция завершается успешно.

11.3 Модификация статьи

11.3.1 Синтаксис модификации статьи

Операция Модификация статьи используется для выполнения одной или нескольких следующих модификаций одиночной статьи:

a) добавление нового атрибута;

b) удаление атрибута;

c) добавление значений атрибута;

d) удаление значений атрибута;

e) замена значений атрибута;

f) модификация псевдонима;

g) добавление константы ко всем значениям атрибута;

h) удаление всех значений атрибута, для которых fallback (возможность восстановления) равна **FALSE** в каждом контексте.

Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Когда затребовано, Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

```
modifyEntry OPERATION  ::=  {
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-modifyEntry )

ModifyEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
        object                  [0]     Name,
        changes                 [1]     SEQUENCE OF EntryModification,
        selection               [2]     EntryInformationSelection OPTIONAL,
        COMPONENTS OF                   CommonArguments } }

ModifyEntryResult ::= CHOICE {
        null                    NULL,
        information             OPTIONALLY-PROTECTED-SEQ {
                SEQUENCE {
                        entry                   [0]     EntryInformation OPTIONAL,
                        COMPONENTS OF           CommonResultsSeq } } }

EntryModification   ::=     CHOICE {
        addAttribute        [0]     Attribute,
        removeAttribute     [1]     AttributeType,
        addValues           [2]     Attribute,
        removeValues        [3]     Attribute,
        alterValues         [4]     AttributeTypeAndValue,
        resetValue          [5]     AttributeType,
        replaceValues       [6]     Attribute }
```

11.3.2 Аргументы модификации статьи

Аргумент **object** определяет статью, к которой должны применяться модификации. Аргумент **object** может быть альтернативным именем и содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **changes** определяет последовательность модификаций, которые должны применяться в указанном порядке. Если любая из отдельных модификаций терпит неудачу, то генерируется уведомление об ошибке **attributeError**, а статья остается в том состоянии, в котором она была до операции. Таким образом, операция является атомарной (неделимой). Конечный результат последовательности модификаций не должен нарушать схему Справочника. Однако возможно, а иногда необходимо, чтобы некоторые изменения **EntryModification** приводили к изменению схемы. Могут выполняться следующие типы модификации:

a) **addAttribute** – Этот аргумент определяет новый атрибут, который следует добавить к статье, полностью определенной аргументом. Любая попытка добавить уже существующий атрибут приведет к **attributeError**.

b) **removeAttribute –** Этот аргумент определяет (своим типом) атрибут, который следует удалить из статьи. Любая попытка удалить несуществующий атрибут приведет к **attributeError**.

ПРИМЕЧАНИЕ 1. – Эта операция не разрешена, если рассматриваемый тип атрибута присутствует в RDN.

c) **addValues –** Этот аргумент определяет атрибут с помощью типа атрибута, указанного в этом аргументе, и определяет одно или более значений атрибута, которые следует добавить к атрибуту. Попытка добавить уже существующее значение приведет к ошибке. Попытка добавить значение к несуществующему типу приведет к добавляемому типу и значению.

d) **removeValues** – Этот аргумент определяет атрибут с помощью типа атрибута, указанного в этом аргументе, и определяет одно или более значений атрибута, которые следует удалить из атрибута. Если значения не присутствуют в атрибуте, то это приводит к **attributeError**. Попытка удалить последнее значение из атрибута приводит к удаляемому типу атрибута.

ПРИМЕЧАНИЕ 2. – Эта операция не разрешена, если одно из этих значений присутствует в RDN.

Атрибуты или значения атрибутов, которые следует добавить, могут быть определены как с контекстным списком, так и без него. Контексты не могут быть добавлены к существующим значениям атрибута, удалены из существующих значений атрибута и не могут быть изменены. Для изменения контекстного списка существующего значения атрибута сначала удаляют это значение атрибута, а затем вставляют то же самое значение атрибута с новым контекстным списком. Когда значение атрибута удалено, контекстный список не должен предоставляться, а любой существующий контекстный список, связанный с удаляемым значением атрибута, удаляется вместе со значением атрибута.

e) **alterValues** – Этот аргумент определяет тип атрибута и определяет некоторую величину, которую следует добавить ко всем значениям этого атрибута. Попытка применения этой модификации к атрибуту, синтаксис которого отличается от **INTEGER** или **REAL**, приводит к **attributeError**.

f) **resetValue –** Этот аргумент определяет атрибут с помощью его типа и удаляет все значения этого атрибута (если они присутствуют), имеющие связанный со значением атрибута контекст, для которого **fallback** (возможность восстановления) равна **FALSE**. Аргумент **resetValue** не удаляет значения атрибута, которые не имеют контекста.

g) **replaceValues** – Этот аргумент заменяет все существующие значения данного типа аргумента подаваемыми значениями, создавая тип атрибута, если он не существовал. Замена без значения удаляет тип атрибута, если он существует, и игнорируется, если тип не существует.

ПРИМЕЧАНИЕ 3. – Данная спецификация Справочника не устанавливает правил в отношении порядка, в котором исполняющий DSA должен декодировать и обрабатывать блоки PDU, которые он получает. Если DSA декодирует весь PDU до обработки каждого элемента и если для нефакультативного параметра CHOICE принимается новое и неожиданное значение, как например **replaceValues**, то возможно, что DSA сообщит об ошибке кодирования. Если, однако, DSA декодирует элементы ввиду необходимости в них, то весьма вероятно, что он обнаружит неизвестное критическое расширение и выдаст код причины неподдерживаемого критического расширения в сигнал о том, что операция не выполнена. В любом случае, для DSA является правильным не обрабатывать операцию; однако разработчики должны быть осведомлены о том, что любой сигнал может использоваться для указания невыполнения операции.

Значения могут быть заменены комбинацией **addValues** и **removeValues** в одной операции Модификация статьи.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Опция **dontDereferenceAlias** игнорируется (и трактуется как установленная), если бит критического расширения **useAliasOnUpdate** не установлен в **criticalExtensions**. Таким образом, псевдонимы переименовываются этой операцией только тогда, когда опция **dontDereferenceAlias** не установлена, а бит **useAliasOnUpdate** установлен. Компонент **sizeLimit** игнорируется (если присутствует). Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

ПРИМЕЧАНИЕ 4. – Операции обновления, которые охватывают переименование имени псевдонима, будут всегда получать отказ, если они наталкиваются на DSA первого издания.

Аргумент **selection** определяет факультативный выбор информации статьи, который указывает, будет ли выдаваться информация в результате операции, и указывает конкретные атрибуты и значения, которые будут выдаваться. Этот аргумент должен определяться только тогда, когда версия, согласованная при операции Привязывание, равна **v2** или выше.

Операция может использоваться для изменения операционных атрибутов Справочника. Могут быть модифицированы только те операционные атрибуты Справочника, которые не отнесены к классу **noUserModification** (и к которым пользователь имеет действующие права доступа к модификации).

ПРИМЕЧАНИЕ 5. – Независимо от того, разрешена пользователю модификация или нет, Справочник может изменять значения операционных атрибутов Справочника из-за побочного влияния других операций Справочника.

Операция может использоваться для модификации совокупных атрибутов только тогда, когда параметр управления службой **subentries** имеет значение **TRUE**, а аргумент **object** является подстатьей, фактически содержащей совокупный(е) атрибут(ы), который(е) следует модифицировать.

ПРИМЕЧАНИЕ 6. – Поэтому при модификации информации, выданной при чтении статьи, должна соблюдаться предосторожность: часть информации может быть получена от совокупных атрибутов и не может быть изменена в операции, применяемой к самой статье. Например, невозможно удалить совокупный атрибут из (обычной) статьи посредством модификации статьи **removeAttribute,** примененной к этой статье (было бы выдано уведомление об ошибке **attributeError** с указанием причины **noSuchAttributeOrValue**).

Операция может использоваться для модификации значения атрибута статьи Класс объекта, если эти значения определяют классы вспомогательных объектов. Однако попытка изменить значение Класс объекта, которое определяет класс структурного объекта статьи, приведет к **updateError** с указанием причины **objectClassModificationProhibited**. Любая модификация классов вспомогательных объектов должна оставить последовательности надклассов непротиворечивыми и соответствующими определению результирующего класса объекта.

11.3.3 Результаты модификации статьи

Если запрос завершается успешно, то должен быть выдан некоторый результат. Если параметр **selection** не был определен в аргументе операции, а результат не должен быть подписан, зашифрован, либо подписан и зашифрован, то выдается нулевой результат. Если параметр **selection** не был определен (но результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником), то компонент статьи опускается. Если результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) параметра **CommonResultsSeq** (см. п. 7.4) включается в результаты. Если результат не должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то никакая информация не должна переноситься с результатом.

11.3.4 Ошибки модификации статьи

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

11.3.5 Точки принятия решения операции Модификация статьи при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **basic-access-control** влияет на модифицируемую статью, то применяется следующая последовательность параметров управлений доступом:

1) Для изменяемой статьи требуется разрешение *Modify (Модификация)*. Если это разрешение не предоставляется, то операция заканчивается неудачей согласно п. 7.11.1.
2) Для каждого из указанных аргументов **EntryModification**, применяемых последовательно, требуются следующие разрешения:
   i) Разрешение *Add (Добавление)* для типа атрибута и для каждого из значений, указанных в параметре **addAttribute**. Если эти разрешения не даны или атрибут уже существует, то операция заканчивается неудачей согласно п. 11.3.5.1 a).
   ii) Разрешение *Remove (Удаление)* для типа атрибута, указанного в параметре **removeAttribute**. Если это разрешение не дано, то операция заканчивается неудачей согласно п. 11.3.5.1 b).

       ПРИМЕЧАНИЕ 1.– Для любого значения атрибута, присутствующего в удаляемом атрибуте, не требуется специального разрешения.

   iii) Разрешение *Add* для каждого значения атрибута, указанного в параметре **addValues**. Если это разрешение не дано или какое-либо из значений атрибута уже существует, то операция заканчивается неудачей согласно п. 11.3.5.1 c).
   iv) Разрешение *Remove* для каждого значения, указанного в параметре **removeValues**. Если это разрешение не дано или какое-либо из значений атрибута уже существует, то операция заканчивается неудачей согласно п. 11.3.5.1 d).

       ПРИМЕЧАНИЕ 2. – Если конечным результатом модификации **removeValues** должно быть удаление последнего значения атрибута (что ведет к удалению самого атрибута), то также требуется разрешение *Remove* для указанного типа атрибута.

   v) Разрешения *Add* и *Remove* для каждого значения, указанного в параметре **alterValues**. Если эти разрешения не даны, то операция заканчивается неудачей согласно п. 11.3.5.1 e).
   vi) Разрешение *Remove* для каждого из значений, которые следует удалить посредством параметра **resetValue**. Если, по крайней мере, одно значение должно быть удалено, а эти разрешения не даны, то операция заканчивается неудачей согласно п. 11.3.5.1 f).

11.3.5.1 Выдача уведомлений об ошибке

Если операция заканчивается неудачей, как определено в п. 11.3.5, то применяется следующая процедура:

a) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт i), то действительно одно из выдаваемых уведомлений об ошибке: когда атрибут уже существует, а разрешение *DiscloseOnError* или *Add* дано для этого атрибута, выдается **attributeError** с указанием причины **attributeOrValueAlreadyExists**; в противном случае выдается **securityError** с указанием причины **insufficient AccessRights** или **noInformation**.

b) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт ii), то действительно одно из выдаваемых уведомлений об ошибке: когда разрешение *DiscloseOnError* дано для удаляемого атрибута и этот атрибут существует, выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**; в противном случае выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

c) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт iii), то действительно одно из выдаваемых уведомлений об ошибке: когда значение атрибута уже существует, а разрешение *DiscloseOnError* или Add дано для этого значения атрибута, выдается **attributeError** с указанием причины **attributeOrValueAlreadyExists**; в остальных случаях проверяется разрешение *DiscloseOnError* на уровне атрибута. Если разрешение *DiscloseOnError* дано для этого атрибута, то выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**; в противном случае выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

d) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт iv), то действительно одно из выдаваемых уведомлений об ошибке: когда разрешение *DiscloseOnError* дано для любого из удаляемых значений атрибута, выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**; в противном случае выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

e) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт v), то действительно одно из выдаваемых уведомлений об ошибке: когда разрешение *DiscloseOnError* дано для любого из изменяемых значений атрибута, выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**; в противном случае выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

f) Если операция заканчивается неудачей согласно п. 11.3.5 2), подпункт vi), то действительно одно из выдаваемых уведомлений об ошибке: когда разрешение *DiscloseOnError* дано для любого из удаляемых значений атрибута, выдается **securityError** с указанием причины **insufficientAccessRights** или **noInformation**; в противном случае выдается **attributeError** с указанием причины **noSuchAttributeOrValue**.

11.3.6 Точки принятия решения операции Модификация статьи при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на изменяемую статью, то применяется следующая последовательность управления доступом:

1) Если разрешение на уровне статьи на основе правила не дано для целевой статьи, то операция заканчивается неудачей и выдает **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

2) Процедура **basic-access-control** применяется на уровне статьи согласно п. 11.3.5.1.

3) Доступ должен быть разрешен к каждому из удаляемых значений атрибута (если оно существует). Если процедура **rule-based-access-control** не дает разрешения для какого-либо значения атрибута, которое следует удалить, то операция заканчивается неудачей и выдает **attributeError** с указанием причины **noSuchAttributeOrValue**.

4) Процедура **basic-access-control** применяется на уровне атрибутов согласно п. 11.3.5 2).

11.4 Модификация Выделенного имени (DN)

11.4.1 Синтаксис модификации DN

Операция Модификация DN используется для изменения Относительно выделенного имени статьи, для изменения первичного Относительно выделенного имени статьи, для добавления и изъятия выделенных значений атрибутов и/или для перемещения статьи в новый порождающий объект DIT. Она может использоваться со статьями объектов, включая составные статьи и статьи псевдонима.

Для членов семейства ее использование ограничивается случаем, когда затронутые члены семейства остаются в той же составной статье.

Если статья имеет подчиненные статьи, то все подчиненные статьи переименовываются или перемещаются соответственно (т. е. поддерево остается неизменным). Аргументы операции могут быть подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Когда затребовано, Справочник может подписать, зашифровать, либо подписать и зашифровать результат.

ПРИМЕЧАНИЕ 1. – Системы первого издания могут использовать эту операцию только для изменения Относительно выделенного имени статьи-листа.

ПРИМЕЧАНИЕ 2. – Системы второго и последующего изданий могут использовать эту операцию для перемещения статей к новой предшествующей статье только тогда, когда старая предшествующая, новая предшествующая, сама статья и все ее подчиненные статьи находятся в одном DSA.

ПРИМЕЧАНИЕ 3. – Операция не перемещает статьи в новый DSA; все статьи остаются в исходном DSA.

ПРИМЕЧАНИЕ 4. – Операция заканчивается успешно или неудачно как единое целое; она не закончится неудачей, если некоторые статьи перемещены, а некоторые не перемещены. Никакие промежуточные состояния операции не должны быть видимы извне пользователям Справочника.

ПРИМЕЧАНИЕ 5. – Могут потребоваться некоторые местные действия после выполнения этой операции, чтобы сохранить совместимость, например, для обновления атрибутов в статьях, которые содержат значения Выделенного имени, относящегося к переименованной или перемещенной статье(ям).

ПРИМЕЧАНИЕ 6. – Атрибут **modifyTimeStamp** не обновляется для статей, подчиненных переименованной или перемещенной статье.

```
modifyDN OPERATION  ::= {
        ARGUMENT        ModifyDNArgument
        RESULT          ModifyDNResult
        ERRORS          { nameError I serviceError I referral ( securityError I updateError }
        CODE            id-opcode-modifyDN }

ModifyDNArgument  ::= OPTIONALLY-PROTECTED {
        SET {
            object              [0]     DistinguishedName,
            newRDN              [1]     RelativeDistinguishedName,
            deleteOldRDN        [2]     BOOLEAN DEFAULT FALSE,
            newSuperior         [3]     DistinguishedName OPTIONAL,
            COMPONENTS OF               CommonArguments} }
ModifyDNResult  ::= CHOICE {
        null                NULL,
        information         OPTIONALLY-PROTECTED-SEQ {
                SEQUENCE {
                        newRDN                  RelativeDistinguishedName,
                        COMPONENTS OF           CommonResultsSeq } } }
```

11.4.2 Аргументы модификации DN

Аргумент **object** идентифицирует статью, Выделенное имя которой должно измениться. Псевдонимы в имени не должны быть переименованы. Аргумент **object** может быть альтернативным именем и содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Аргумент **newRDN** указывает новое RDN статьи. Если операция перемещает статью в новую предшествующую статью без изменения ее RDN, то для этого параметра применяется старое RDN.

Если какое-либо значение атрибута в новом RDN еще не существует в статье (как часть старого RDN или как невыделенное значение), то оно добавляется. Если оно не может быть добавлено, то выдается уведомление об ошибке.

Для каждого атрибута, дающего вклад в RDN, аргумент **newRDN** может дать альтернативные выделенные значения, если эти выделенные значения различаются контекстом, как описано в 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если это так, то аргумент **newRDN** должен быть первичным RDN и должен содержать все выделенные значения с их контекстными списками для всех атрибутов, дающих вклад в RDN (включая существующие выделенные значения, которые должны быть сохранены в качестве выделенных значений). Параметр **AttributeTypeAndDistinguishedValue** в **newRDN**, который дается без альтернативных выделенных значений, указывает одиночное выделенное значение для этого атрибута.

Если установлен признак **deleteOldRDN**, то все значения атрибута в старом RDN, которые не присутствуют в новом RDN, удаляются. Это применимо также к альтернативным выделенным значениям с отличающимися контекстами, если они существуют в старом RDN, но не включены в новое RDN. Если этот признак не установлен, то старые выделенные значения должны остаться в статье (но далее не как выделенные значения). Этот признак должен быть установлен, когда атрибут с одним значением в RDN имеет значение, измененное операцией. Если значение атрибута в старом RDN такое же, как в новом RDN, без учета их контекстных списков, то значение в старом RDN заменяется значением из нового RDN. Если эта операция удаляет последнее значение в атрибуте, то этот атрибут удаляется.

Аргумент **newSuperior**, если он присутствует, указывает на перемещение статьи к новой предшествующей статье в DIT. Статья становится непосредственно подчиненной статье с указанным Выделенным именем, которая должна быть уже существующей статьей объекта. Новая предшествующая статья не должна быть самой рассматриваемой статьей или одной из ее подчиненных статей, либо псевдонимом, либо такой статьей, что перемещенная статья нарушит какие-либо структурные правила DIT. Возможно, что статьи, *подчиняющиеся* перемещенной статье, могут нарушить действующую подсхему, тогда Административный орган подсхемы является ответственным за последующие корректировки этих статей, чтобы сделать их совместимыми с подсхемой, как описано в разделе 14 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Если этот аргумент присутствует, то в параметре **criticalExtensions** информации **CommonArguments** должен быть установлен бит **newSuperior**, указывающий, что это расширение является критическим.

Аргумент **newSuperior** может быть альтернативным именем и может содержать контекстную информацию, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Информация **CommonArguments** (см. п. 7.3) содержит спецификацию параметров управления службой и параметров безопасности, применяемых к запросу. Для целей этой операции опция **dontDereferenceAlias** и компонент **sizeLimit** не применимы, а если они присутствуют, то игнорируются. Псевдонимы никогда не переименовываются этой операцией. Если аргумент этой операции должен быть подписан, зашифрован, либо подписан и зашифрован инициатором, то компонент **SecurityParameters** (см. п. 7.10) включается в аргументы.

11.4.3 Результаты модификации DN

Если запрос завершается успешно, то должен быть выдан некоторый результат. Если этот результат должен быть подписан, зашифрован, либо подписан и зашифрован Справочником, то компонент **SecurityParameters** (см. п. 7.10) параметра **CommonResultsSeq** (см. п. 7.4) и новый RDN включаются в результаты. Если результат операции не должен быть подписан Справочником, то никакая информация не должна переноситься с результатом.

11.4.4 Ошибки модификации DN

Если запрос закончился неудачей, то должно быть сообщено одно из перечисленных уведомлений об ошибке. Обстоятельства, при которых должны быть сообщены конкретные уведомления об ошибке, определяются в разделе 12.

11.4.5 Точки принятия решения операции Модификация DN при управлении базовым доступом

Если применяется также процедура **rule-based-access-control**, то порядок ее применения наряду с процедурой **basic-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **basic-access-control** влияет на статью, подлежащую переименованию, то применяются следующие параметры управления доступом:

- Если результатом операции является изменение RDN статьи, то требуется разрешение *Rename* (*Переименование*) для статьи, подлежащей переименованию (рассматриваемой с ее первоначальным именем). Если это разрешение не дано, то операция заканчивается неудачей согласно п. 11.4.5.1.
- Если результатом операции является перемещение статьи к новой предшествующей статье в DIT, то требуется разрешение *Export* (*Экспорт*) для статьи, рассматриваемой с ее первоначальным именем, и требуется разрешение *Import* (*Импорт*) для статьи, рассматриваемой с ее новым именем. Если любое из этих разрешений не дано, то операция заканчивается неудачей согласно п. 11.4.5.1.

ПРИМЕЧАНИЕ 1. – Разрешение *Import* должно обеспечиваться, как предписывает ACI (Access Control Information, информация управления доступом).

ПРИМЕЧАНИЕ 2. – Никакие дополнительные разрешения не требуются даже в случае, когда в результате изменения в имени последнего RDN новое выделенное значение должно быть добавлено или старое значение удалено.

11.4.5.1 Выдача уведомлений об ошибке

Если операция заканчивается неудачей, как определено в п. 11.4.5, то выполняется процедура, описанная в п. 7.11.1, применительно к статье, подлежащей переименованию (рассматриваемой с ее первоначальным именем).

11.4.6 Точки принятия решения операции Модификация DN при управлении доступом на основе правила

Если применяется также процедура **basic-access-control**, то порядок ее применения наряду с процедурой **rule-based-access-control** является местным вопросом, за исключением того, что если отклонен доступ к статье, типу атрибута или значению атрибута каким-либо механизмом, то это не должно отменяться другим механизмом. Например, разрешение *DiscloseOnError* от процедуры **basic-access-control** является разрешением, которое не должно отменять запрет от процедуры **rule-based-access-control**.

Если процедура **rule-based-access-control**, или **rule-and-basic-access-control**, или **rule-and-simple-access-control** влияет на статью, подлежащую переименованию, то применяется следующая последовательность управления доступом:

1) Если разрешение для RDN на основе правила не дано целевой статье, то операция заканчивается неудачей и выдает **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.
2) Процедура **basic-access-control** применяется на уровне статьи согласно п. 11.4.5.
3) Если результатом операции является перемещение статьи к новой предшествующей статье в DIT, то на основе правила требуется разрешение на RDN для новой предшествующей статьи, в противном случае операция заканчивается неудачей и выдает **nameError** с указанием причины **noSuchObject** согласно п. 7.11.2.4.

12 Ошибки

12.1 Приоритет ошибки

Справочник прекращает выполнение операции, когда он определил, что должно быть передано уведомление об ошибке.

ПРИМЕЧАНИЕ 1. – Из этого правила следует, что первая встреченная ошибка может отличаться от последующих ошибок при повторных обращениях с тем же запросом, так как не имеется определенного логического порядка обработки полученного запроса. Например, обращения к разным DSA могут выполняться в различном порядке.

ПРИМЕЧАНИЕ 2. – Правила приоритета ошибки, указанные здесь, применяются только к абстрактной службе, обеспеченной Справочником в целом. Что касается внутренней структуры Справочника, то применяются различные правила.

Если Справочник одновременно обнаруживает более одной ошибки, то нижеследующий перечень определяет ошибку, о которой сообщается. Ошибка, расположенная выше в перечне, имеет более высокий логический приоритет, чем находящаяся ниже, и является ошибкой, о которой сообщается.

a) **nameError**;

b) **updateError**;

c) **attributeError**;

d) **securityError**;

e) **serviceError**.

Следующие ошибки не создают конфликтов приоритета:

a) **abandonFailed** (невыполненный отказ), так как эта ошибка специфична для одной операции Отказ, не имеющей других ошибок;

b) **abandoned** (отказано), о которой не сообщается, если операция Отказ начата одновременно с обнаружением какой-либо ошибки. В этом случае выдается уведомление об ошибке **abandonFailed** с указанием причины **tooLate**, вместе с сообщением о фактической выявленной ошибке;

c) **referral** (отсылка)**,** которая не является "настоящей" ошибкой, а только указанием на то, что Справочник обнаружил, что DUA должен представить свой запрос в другой пункт доступа.

12.2 Отказано

Этот результат может быть сообщен для любой незавершенной операции запроса Справочника (т. е. Чтение, Поиск, Сравнение, Список), если DUA запросил операцию Отказ с соответствующим аргументом **Invokeld**. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
abandoned ERROR ::=     { -- не является фактически "ошибкой"
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {COMPONENTS OF      CommonResults} }
        CODE            id-errcode-abandoned }
```

Компонент **SecurityParameters** (см. 7.10) включается в **CommonResults** (см. 7.4), если ошибка должна быть подписана, зашифрована, либо подписана и зашифрована Справочником.

12.3 Невыполненный отказ

Уведомление об ошибке **abandonFailed** сообщает о проблеме, встреченной во время попытки отказа от какой-либо операции. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, либо подписать и зашифровать параметры ошибки.

```
abandonFailed ERROR ::= {
        PARAMETER               OPTIONALLY-PROTECTED {
                SET {
                        problem         [0]     AbandonProblem,
                        operation       [1]     InvokeId,
                        COMPONENTS OF CommonResults } }
        CODE                    id-errcode-abandonFailed }

AbandonProblem ::= INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }
```

Различные параметры имеют следующий смысл.

Параметр **problem** указывает конкретную встреченную проблему. Может быть указана любая из следующих причин:

a) **noSuchOperation –** если Справочнику не известно об операции, от выполнения которой следует отказаться (это может быть, если запроса такой операции не было или если Справочник забыл о нем);

b) **tooLate** – когда Справочник уже выдал результат операции;

c) **cannotAbandon** – если сделана попытка отменить операцию, для которой это запрещено (например, для модификации), или если отказ не может быть выполнен.

Параметр **operation** указывает конкретную операцию (вызов), которая должна быть отменена.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если ошибка должна быть подписана, зашифрована, либо подписана и зашифрована Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification** в **CommonResults**.

12.4 Ошибка атрибута

Компонент **attributeError** сообщает о проблеме, связанной с атрибутом. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
attributeError ERROR  ::=  {
        PARAMETER     OPTIONALLY-PROTECTED {
                SET {
                        object          [0]     Name,
                        problems        [1]     SET OF SEQUENCE {
                                problem         [0]     AttributeProblem,
                                type            [1]     AttributeType,
                                value           [2]     AttributeValue OPTIONAL},
                        COMPONENTS OF CommonResults } }
        CODE                    id-errcode-attributeError }

AttributeProblem ::= INTEGER {
        noSuchAttributeOrValue                  (1),
        invalidAttributeSyntax                  (2),
        undefinedAttributeType                  (3),
        inappropriateMatching                   (4),
        constraintViolation                     (5),
        attributeOrValueAlreadyExists           (6),
        contextViolation                        (7) }
```

Различные параметры имеют следующий смысл.

Параметр **object** указывает статью, к которой применялась операция, когда произошла ошибка. Выдаваемое имя может содержать только первичные выделенные значения для атрибутов, содержащих несколько выделенных значений, различающихся контекстами (т. е. DSA не должен применять контекстный выбор, описанный в п. 7.7, как он это делает для успешных операций).

Может быть указана одна или несколько причин (проблем). Каждая причина (указанная ниже) сопровождается указанием на тип атрибута (параметр **type**), а в случае необходимости устранения неоднозначности – на значение (параметр **value**), которое вызвало ошибку:

a) **noSuchAttributeOrValue** – Указанная статья не содержит один из атрибутов или одно из значений атрибута, указанные в качестве аргумента операции.

b) **invalidAttributeSyntax** – Потенциальное значение атрибута, указанное в качестве аргумента операции, не соответствует синтаксису атрибута данного типа.

c) **undefinedAttributeType** – В качестве аргумента операции указан не определенный ранее тип атрибута. Эта ошибка может появиться только при операциях Добавление статьи и Модификация статьи.

d) **inappropriateMatching** – Сделана попытка, например, в фильтре, использовать правило сопоставления, не определенное для рассматриваемого типа атрибута.

e) **constraintViolation** – Значение атрибута, заданное в аргументе операции, не удовлетворяет ограничениям, наложенным в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2 или в определении атрибута (например, значение превышает максимально допустимый размер).

f) **attributeOrValueAlreadyExists** – Сделана попытка добавления атрибута, который уже существует в статье, или значения, которое уже существует в атрибуте.

g) **contextViolation** – Контекстный список или контекст, представленный со значением атрибута в аргументе операции, не удовлетворяет ограничениям, наложенным в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2, в определении контекста (например, контекстное значение имеет неправильный синтаксис), либо изложенным в DIT Context Use (Использование контекста DIT).

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если уведомление об ошибке должно быть подписано, зашифровано, либо подписано и зашифровано Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification** в **CommonResults**.

12.5 Ошибка имени

Компонент **nameError** сообщает о проблеме, связанной с именем, заданным в качестве аргумента операции. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
nameError ERROR ::= {
        PARAMETER    OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]    NameProblem,
                        matched             [1]    Name,
                        COMPONENTS OF       CommonResults } }
        CODE                id-errcode-nameError }

NameProblem ::= INTEGER {
        noSuchObject                    (1),
        aliasProblem                    (2),
        invalidAfributeSyntax           (3),
        aliasDereferencingProblem       (4),
        contextProblem                  (5) }
```

Различные параметры имеют следующий смысл.

Параметр **problem** указывает конкретную встреченную проблему. Может быть указана любая из следующих причин:

a) **noSuchObject** – Представленное имя не соответствует имени какого-либо объекта.

b) **aliasProblem** – Псевдоним был переименован с указанием имени несуществующего объекта.

c) **invalidAttributeSyntax** – Тип атрибута и сопровождающее его значение атрибута, указанные в имени в процедуре AVA, несовместимы.

d) **aliasDereferencingProblem** – Псевдоним встречен в ситуации, где он не допустим или где доступ запрещен.

e) **contextProblem** – Тип или значение контекста, использованные в имени, не поняты или недействительны, либо использование контекстного варианта имени неприемлемо, либо в процессе распознавания имени потенциальное имя соответствует именам нескольких статей DIT.

Параметр **matched** содержит имя низшей статьи (объекта или псевдонима) в DIT, для которой было выполнено сопоставление и которая представляет собой усеченную форму предоставляемого имени или, если псевдоним был переименован, окончательного имени. Выдаваемое имя может содержать только первичные выделенные значения для атрибутов, содержащих несколько выделенных значений, различающихся контекстами (т. е. DSA не должен применять контекстный выбор, описанный в 7.7, как он это делает для успешных операций).

ПРИМЕЧАНИЕ. – Если есть проблема с типами атрибута и/или со значениями в имени, представленном в аргументе какой-либо операции Справочника, то об этом сообщается посредством **nameError** с указанием причины **invalidAtributeSyntax**, а не посредством **attributeError** или **updateError**.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если ошибка должна быть подписана, зашифрована, либо подписана и зашифрована Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification** в **CommonResults**.

12.6 Отсылка

Компонент **referral** перенаправляет пользователя службы в один или несколько пунктов доступа, которые лучше оборудованы для выполнения запрашиваемой операции. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
referral ERROR ::= {  -- не является фактически "ошибкой"
        PARAMETER OPTIONALLY-PROTECTED {
                SET {
                        candidate               [0]  ContinuationReference,
                        COMPONENTS OF                CommonResults } }
        CODE                        id-errcode-referral }
```

Уведомление об ошибке имеет единственный параметр, содержащий **ContinuationReference**, который может быть использован для продолжения операции (см. Рекомендацию МСЭ-Т X.518 | ИСО/МЭК 9594-4).

Если DSA отвечает на запрос LDAP, то компонент **accessPoints** в **ContinuationReference** должен содержать действительное значение **LabeledURI**, в случае чего он будет использовать эту величину для создания отсылки LDAP. Если он не содержит действительного значения **LabeledURI**, он не должен выдавать отсылку.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если уведомление об ошибке должно быть подписано Справочником.

Перед операцией с отсылкой на продолжение DUA должен проверить, что идентичный запрос, который был бы сгенерирован этой отсылкой на продолжение, не был уже выдан как часть обработки этого же запроса пользователя. Если это имеет место, то DUA не должен работать с отсылкой на продолжения. Это позволяет избежать циклических повторений.

12.7 Ошибка безопасности

Параметр **securityError** указывает на проблему, связанную с аспектами безопасности при выполнении операции. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
securityError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]     SecurityProblem,
                        spkmInfo            [1]     SPKM-ERROR,
                        COMPONENTS OF               CommonResults } }
        CODE                        id-errcode-securityError }

SecurityProblem ::= INTEGER {
        inappropriateAuthentication     (1),
        invalidCredentials              (2),
        insufficientAccessRights        (3),
        invalidSignature                (4),
        protectionRequired              (5),
        noInformation                   (6),
        blockedCredentials              (7),
        invalidQOPMatch                 (8),
        spkmError                       (9) }
```

Уведомление об ошибке имеет единственный параметр, который сообщает о конкретной встреченной проблеме. Могут быть указаны следующие причины:

a) **inappropriateAuthentication** – Уровень безопасности, связанной с удостоверением инициатора, несовместим с требуемым уровнем безопасности, например, были представлены простые удостоверения, в то время как требовались строгие удостоверения.

b) **invalidCredentials** – Представленные удостоверения недействительны.

c) **insufficientAccessRights** – Инициатор не имеет права выполнять запрошенную операцию.

d) **invalidSignature** – Обнаружено, что подпись в запросе недействительна.

e) **protectionRequired** – Справочник не пожелал выполнить запрошенную операцию, так как аргумент не подписан.

f) **noInformation** – Выполнение запрошенной операции привело к ошибке безопасности, для которой нет подходящей информации.

g) **blockedCredentials** – Удостоверения блокированы по соображениям безопасности (например, недействительный пароль был последовательно представлен слишком много раз). Решение о выдаче этой причины определяется действующей стратегией безопасности для DSA.

h) **invalidQOPMatch** – Два объекта имеют отличающиеся параметры защиты, определенные для соответствующих служб безопасности.

i) **spkmError** – Было обнаружено, что представленный маркер SPKM недействителен. Параметр **spkmInfo** содержит указание, что это является ошибочным маркером SPKM, и идентификатор контекста SPKM, с которым связана эта ошибка.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если уведомление об ошибке должно быть подписано, зашифровано, либо подписано и зашифровано Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification** в **CommonResults**.

12.8 Ошибка службы

Параметр **serviceError** указывает на проблему, связанную с обеспечением службы. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
serviceError ERROR ::= {
    PARAMETER       OPTIONALLY-PROTECTED {
        SET{
            problem             [0]    ServiceProblem,
            COMPONENTS OF CommonResults } }
    CODE                        id-errcode-serviceError }

ServiceProblem ::= INTEGER {
    busy                                (1),
    unavailable                         (2),
    unwillingToPerform                  (3),
    chainingRequired                    (4),
    unableToProceed                     (5),
    invalidReference                    (6),
    timeLimitExceeded                   (7),
    administrativeLimitExceeded         (8),
    loopDetected                        (9),
    unavailableCriticalExtension        (10),
    outOfScope                          (11),
    ditError                            (12),
    invalidQueryReference               (13),
    requestedServiceNotAvailable        (14),
    unsupportedMatchingUse              (15) }
    ambiguousKeyAttributes              (16),
    saslBindInProgress                  (17) }
```

Уведомление об ошибке имеет единственный параметр, который сообщает о конкретной встреченной проблеме. Могут быть указаны следующие причины:

a) **busy** – Справочник либо некоторая его часть в настоящее время слишком занят, чтобы выполнить запрошенную операцию, но он сможет выполнить ее через короткий промежуток времени.

b) **unavailable** – Справочник либо некоторая его часть в настоящее время недоступен.

c) **unwillingToPerform** – Справочник либо некоторая его часть не готов к выполнению этого запроса, например потому, что это привело бы к чрезмерному использованию ресурсов или к нарушению стратегии, установленной Административным органом.

d) **chainingRequired** – Справочник не может выполнить запрос без использования сцепления; однако сцепление запрещено опцией управления службой **chainingProhibited**.

e) **unableToProceed** – DSA, выдающий это уведомление об ошибке, не имеет административных полномочий для управления соответствующим контекстом именования и поэтому не может участвовать в проверке имени.

f) **invalidReference** – DSA не может выполнить запрос, направленный из DUA (с помощью **OperationProgress**). Это могло возникнуть из-за использования недействительной отсылки.

g) **timeLimitExceeded** – Справочник исчерпал лимит времени, установленный пользователем в параметре управления службой. Нет частичного результата, доступного для выдачи пользователю.

h) **administrativeLimitExceeded** – Справочник достиг некоторого лимита, установленного административным органом, и нет частичного результата, доступного для выдачи пользователю.

i) **loopDetected** – Справочник не может выполнить этот запрос из-за внутреннего зацикливания.

j) **unavailableCriticalExtension** – Справочник не может выполнить запрос, так как одно или несколько критических расширений оказались недоступными.

k) **outOfScope** – Никакие отсылки не доступны в пределах запрошенной области.

l) **ditError** - Справочник не может выполнить запрос из-за проблемы совместимости с DIT.

m) **invalidQueryReference** – Параметры запрошенной операции недействительны. Об этой причине сообщается, если параметр **queryReference** в постраничных результатах является недействительным.

ПРИМЕЧАНИЕ. – Эта причина не поддерживается системами первого издания.

n) **requestedServiceNotAvailable** – Запрос поиска завершился неудачей в пределах зависящей от службы административной области, потому что нет доступного правила поиска для операции Поиск или потому, что поиск нарушил применяемое правило поиска. Вместе с этой ошибкой службы может быть выдана дополнительная диагностическая информация. Такая дополнительная информация для различных ситуаций определяется в разделе 13.

o) **unsupportedMatchingUse** – Сделана попытка, например, в фильтре, использовать правило сопоставления, не поддерживаемое DSA, когда установлена опция поиска **performExactly**.

p) **ambiguousKeyAttributes** – Выбрано правило сопоставления на основе отображения, но отображаемые элементы фильтра дают несколько соответствий с имеющейся таблицей отображения. Эта ошибочная ситуация сопровождается атрибутом уведомления согласно действующему правилу сопоставления на основе соответствия.

q) **saslBindInProgress** – Для некоторых механизмов аутентификации инициатору может быть необходимо вызвать несколько раз операцию **directoryBind**. Это указывается отвечающим элементом, направляющим **serviceError** с проблемой **saslBindInProgress**. Это указывает на то, что отвечающий элемент требует у инициатора вызвать операцию **directoryBind** с тем же самым механизмом **SaslCredentials** для продолжения процесса аутентификации. Если на любом этапе инициатор желает прервать процесс, он может вызвать операцию **directoryBind** с **SaslAbort**, установленным на **TRUE**.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если уведомление об ошибке должно быть подписано, зашифровано, либо подписано и зашифровано Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification в CommonResults**.

12.9 Ошибка обновления

Параметр **updateError** указывает на проблемы, связанные с попытками добавления, удаления или изменения информации в DIB. Если параметры операции были подписаны, зашифрованы, либо подписаны и зашифрованы инициатором (см. п. 17.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2), то Справочник может подписать, зашифровать, либо подписать и зашифровать параметры ошибки.

```
updateError ERROR ::= {
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem             [0]    UpdateProblem,
            attributeInfo       [1]    SET SIZE (1..MAX) OF CHOICE {
                attributeType              AttributeType,
                attribute                  Attribute } OPTIONAL,
            COMPONENTS OF               CommonResults } }
    CODE                        id-errcode-updateError }

UpdateProblem ::= INTEGER {
    namingViolation                 (1),
    objectClassViolation            (2),
    notAllowedOnNonLeaf             (3),
```

notAllowedOnRDN (4),
entryAlreadyExists (5),
affectsMultipleDSAs (6),
objectClassModificationProhibited (7),
noSuchSuperior (8),
notAncestor (9),
parentNotAncestor (10),
hierarchyRuleViolation (11),
familyRuleViolation (12) }

Параметр **problem** сообщает о конкретной встреченной проблеме. Могут быть указаны следующие причины:

a) **namingViolation** – Попытка добавления или модификации нарушила бы структурные правила DIT, определенные в схеме Справочника и в Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2. Другими словами, статья была бы размещена как подчиненная к статье псевдонима или в области DIT, не разрешенной для члена ее класса объектов, либо для статьи было бы определено RDN для включения запрещенного типа атрибута.

b) **objectClassViolation** – Попытка обновления привела бы к созданию статьи, несовместимой с правилами для содержания статьи; например, с определением ее класса объектов, с правилами о содержании DIT или с определениями из Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2, которые касаются классов объектов.

c) **notAllowedOnNonLeaf** – Запрошенная операция разрешена только для статей-листьев DIT.

d) **notAllowedOnRDN** – Запрошенная операция затронула бы RDN (например, удаление атрибута, который является частью RDN).

e) **entryAlreadyExists** – В запрошенной операции **addEntry** или **modifyDN** названа статья, которая уже существует.

ПРИМЕЧАНИЕ 1. – Эта причина охватывает и конфликт, вызванный именами RDN, которые содержат несколько выделенных значений, различающихся контекстами, независимо от контекста, как описано в п. 9.3 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2.

f) **affectsMultipleDSAs** – Попытка обновления вызывает необходимость работать с группой DSA, в которых эта операция не разрешена.

g) **objectClassModificationProhibited** – Операция попыталась изменить структурный класс объектов в статье.

h) **noSuchSuperior** – Предпринятая операция Модификация DN называет новую предшествующую статью, которая не существует.

i) **notAncestor** – Операция попыталась удалить составную статью, не указав порождающий объект.

j) **parentNotAncestor** – Операция попыталась установить статью в качестве иерархически непосредственно порожденного объекта под членом семейства, который не является порождающим объектом.

k) **hierarchyRuleViolation** – Операция попыталась нарушить правило, применимое к иерархической группе: иерархическая группа должна быть либо полностью вне какой-либо зависящей от службы административной области, либо должна полностью находиться в пределах зависящей от службы административной области; иерархическая группа ограничивается одним DSA.

l) **familyRuleViolation** – Операция попыталась нарушить правило, применимое к семействам в пределах составной статьи.

Параметр **attributeInfo** указывает конкретный тип(ы) атрибута и, возможно, значение(я), вызывающие проблемы. Если причина **objectClassViolation** сообщается, то элемент **attribute** должен присутствовать, указывая тип атрибута **objectClass** и перечисляя класс(ы) объектов, которые вызвали проблему; дополнительные элементы **attributeType** также могут присутствовать (например, чтобы указать отсутствующие обязательные или лишние атрибуты).

ПРИМЕЧАНИЕ 2. – Параметр **updateError** не используется для указания проблем с типами атрибута, значениями или с нарушениями ограничений, встречающихся в операциях Добавление статьи, Удаление статьи, Модификация статьи и Модификация DN. Об этих проблемах сообщается посредством **attributeError**.

Компонент **SecurityParameters** (см. п. 7.10) включается в **CommonResults** (см. п. 7.4), если уведомление об ошибке должно быть подписано, зашифровано, либо подписано и зашифровано Справочником.

Информация, возникшая в связи с причиной ошибки, может быть факультативно сообщена при помощи компонента **notification** в **CommonResults**.

13 Анализ аргументов поиска

Данный раздел применим только к операции Поиск, которая начинает свою начальную фазу оценки в пределах некоторой зависящей от службы административной области.

Эта процедура имеет две цели:

a) Она обеспечивает функцию проверки поиска (см. п. 16.12 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2). Однако функция проверки поиска не дает информацию об ошибке. Если во время выполнения этой процедуры встречается ошибка, то оценка прекращается и выдается значение FALSE; в противном случае выдается значение TRUE. Проверка поиска для пустого правила поиска всегда выдает TRUE.

b) Эту процедуру следует использовать, когда никакого управляющего правила поиска не может быть найдено и когда можно определить единственное правило поиска; **SearchArgument** может быть оценен для определения причины неудачи запроса поиска. Если в этом случае обнаружена ошибочная ситуация, то оценка прекращается, предоставляется необходимая диагностическая информация в компоненте **notification** в типе данных **CommonResults** и выдается уведомление об ошибке службы с указанием причины **requestedServiceNotAvailable**. Какая диагностическая информация включается в уведомление, зависит от типа найденной ошибки.

ПРИМЕЧАНИЕ. – Согласно вышеуказанной спецификации запрос поиска может быть оценен дважды по одному правилу поиска. Метод оптимизации этого не является частью данной спецификации, а зависит от реализации.

В процедуре предполагается, что реализация не будет позволять вызываемому правилу поиска:

- указывать неподдерживаемые типы атрибутов, контекстные типы, правила сопоставления, ограничения для сопоставления и т. д.;
- указывать алгоритмы сопоставления на основе отображения, которые являются неподдержанными или неподходящими для типа поиска, которым управляет правило поиска;
- указывать подстановки правила сопоставления, которые нарушали бы правило поиска;
- ссылаться на факультативные свойства правила поиска, которые не поддерживаются реализацией; или
- быть противоречивым или ошибочным.

13.1 Общая проверка фильтра поиска

Первая проверка должна определить, не нарушает ли фильтр некоторые основные ограничения; она выполняется с помощью следующей процедуры:

1) Если имеются типы атрибута, представленные в фильтре, но не представленные в каком-либо профиле атрибутов запроса в компоненте **inputAttributeTypes** правила поиска, то компонент **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchAttributeViolation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **attributeTypeList**, имеющий в качестве значения идентификаторы объектов, определяющие неразрешенные типы атрибута.
2) Если имеются типы атрибута, представленные только инвертированными элементами фильтра, то компонент **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-attributeNegationViolation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **attributeTypeList**, где значениями являются идентификаторы объектов, определяющие типы атрибута, недопустимо инвертированные в фильтре.
3) Проверяется, что условие, указанное в **attributeCombination**, выполняется для неинвертированных имеющихся типов атрибутов. Если обязательные типы атрибута, т. е. типы атрибутов, которые безусловно должны быть представлены неинвертированными элементами в фильтре, отсутствуют в каком-либо подфильтре, то компонент **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-missingSearchAttribute**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **attributeTypeList**, имеющий в качестве значения идентификаторы объектов, определяющие отсутствующие типы атрибута.

   Если требуемая комбинация отсутствует, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchAttributeCombination Violation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **attributeCombinations**, указывающий отсутствующую(ие) комбинацию(и).

4) Для профилей атрибутов запроса, имеющих подкомпонент **selectedValues** с пустым множеством значений, проверяется, имеется ли элемент фильтра для тех типов атрибута, которые не выполняют одно из следующих требований:
   - элемент фильтра имеет тип **present**, а подкомпонент **contexts** отсутствует в профиле атрибутов запроса; или
   - элемент фильтра имеет тип **contextPresent**, а подкомпонент **contexts** присутствует в профиле атрибутов запроса.

Если вышеуказанная проверка завершилась неуспешно для какого-либо элемента фильтра, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchValueNotAllowed**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **filterItem**, имеющий в качестве значений элементы фильтров, которые были неуспешными.

5) Для профилей атрибутов запроса, имеющих подкомпонент **contexts**, проверяется, имеются ли элементы фильтра, которые ссылаются на контекстные типы, не включенные в этот подкомпонент. Если это так, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchContextViolation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **contextTypeList**, имеющий в качестве значений идентификаторы объектов, определяющие неразрешенные контекстные типы.

6) Если в правиле поиска сделан выбор **allowed** для компонента **subset**, то проверяется, удовлетворяет ли аргумент **subset в SearchArgument** этой спецификации. Если нет, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchSubsetViolation**; и
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения.

13.2 Проверка профилей атрибутов запроса

Если вышеупомянутая процедура не дала какую-либо ошибку, то для каждого подфильтра следует проверить, что каждый тип атрибута, представленный в этом подфильтре, имеется фактически. Данная процедура не определяет порядок, в котором подфильтры должны оцениваться. Чтобы тип атрибута фактически присутствовал в подфильтре, он должен быть представлен, по крайней мере, одним неинвертированным элементом фильтра, который подчиняется соответствующему профилю атрибутов запроса. Неинвертированный элемент фильтра оценивается с помощью процедуры, приведенной ниже.

Неинвертированные элементы фильтра проверяются в следующем порядке:

1) элементы фильтра для типов атрибута, которые безусловно должны быть представлены, проверяются для каждого подфильтра;
2) элементы фильтра для типов атрибута, которые должны быть представлены при определенных условиях, проверяются для каждого подфильтра; и
3) оставшиеся элементы фильтра проверяются для каждого подфильтра.

Если подфильтр не проходит оценку успешно, то оценка останавливается, а информация об ошибке выдается, как описано ниже.

Если тип атрибута в подфильтре представлен несколькими неинвертированными элементами фильтра, то каждый такой элемент фильтра, в принципе, проверяется, пока не будет найден подчиняющийся элемент фильтра или не будут проверены все элементы фильтра. Если проверка элемента фильтра терпит неудачу во время этой процедуры, то он не учитывается для дальнейшей оценки. Самый последний элемент фильтра, у которого процедура проверки для типа атрибута потерпела неудачу, определяет выдаваемую диагностическую информацию.

Элемент фильтра оценивается с помощью следующей процедуры:

1) Если компонент **selectedValues** в профиле атрибутов запроса отсутствует, либо если он присутствует и пуст, то проверяется, имеет ли элемент фильтра тип **equality**, **substrings**, **approximateMatch** или **extensibleMatch**. Если нет, то компонент **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchValueRequired**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
   - атрибут уведомления **attributeTypeList**, имеющий в качестве значения идентификатор объекта, определяющий тип атрибута у элемента фильтра.
2) Если подкомпонент **selectedValues** в соответствующем профиле атрибутов запроса присутствует и он не пуст, то проверяется, будет ли элемент фильтра соответствовать любому значению, указанному в этом подкомпоненте. Если это не так, то компонент **notification** должен содержать:

- атрибут уведомления **searchServiceProblem** со значением **id-pr-invalidSearchValue**;
- атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и
- атрибут уведомления **filterItem**, содержащий в качестве единственного значения элемент фильтра, проверка которого была неуспешной.

3) Если подкомпонент **contexts** отсутствует, то происходит переход к следующему подпункту.

4) Проверяется, что условие, указанное в подкомпоненте **contextCombination**, выполняется в части присутствия контекстных типов. Если обязательные контекстные типы, т. е. контекстные типы, которые безусловно должны быть представлены для типа атрибута, отсутствуют, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-missingSearchContext**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения;
   - атрибут уведомления **attributeTypeList**, содержащий в качестве единственного значения идентификатор объекта, определяющий тип атрибута у элемента фильтра; и
   - атрибут уведомления **contextTypeList** с идентификаторами объектов, определяющими отсутствующие контекстные типы.

   Если требуемая комбинация отсутствует, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchContextCombination Violation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения;
   - атрибут уведомления **attributeTypeList**, имеющий в качестве единственного значения идентификатор объекта, определяющий тип атрибута у элемента фильтра; и
   - атрибут уведомления **contextCombinations**, определяющий отсутствующую(ие) комбинацию(и).

5) Проверяется, все ли проверки контекста для рассматриваемого типа атрибута в подфильтре включены в подкомпонент **contexts**. Если это не так, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchContextViolation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения;
   - атрибут уведомления **attributeTypeList**, имеющий в качестве единственного значения идентификатор объекта, определяющий тип атрибута у элемента фильтра; и
   - атрибут уведомления **contextTypeList**, имеющий в качестве значений идентификаторы объектов, определяющие контекстные типы, не разрешенные для этого типа атрибута.

6) Если контекстные значения включены для всех контекстных типов в подкомпонент **contexts** профиля атрибутов запроса, то проверяется, содержит ли какая-нибудь проверка контекста, указанная для типа атрибута в подфильтре, значения, не указанные для соответствующих контекстных типов в подкомпоненте **contexts**. Если это так, то компонент **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-searchContextValueViolation**;
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения;
   - атрибут уведомления **attributeTypeList**, имеющий в качестве единственного значения идентификатор объекта, определяющий тип атрибута у элемента фильтра; и
   - атрибут уведомления **contextList**, имеющий в качестве значений проверки контекстов, не разрешенные для этого типа атрибута.

13.3 Проверка выбора параметров управления и иерархии

Если при поисковом запросе неуспешно завершилось тестирование выборов параметров управления и иерархии, как определено в 16.10.5 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2, то выполняется процедура данного подраздела.

1) Если компонент **defaultControls** правила поиска или подкомпонент **hierarchyOptions** компонента **defaultControls** отсутствует, а поисковый запрос определяет выборы иерархии (кроме **self**), то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-hierarchySelectForbidden**; и
   - атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения.

2) Если в запросе имеются опции выбора иерархии, которые не разрешены, или некоторые выборы отсутствуют в комбинации компонентов **defaultControls** и **mandatoryControls** правила поиска, то **notification** должен содержать:
   - атрибут уведомления **searchServiceProblem** со значением **id-pr-invalidHierarchySelect**;

– атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и

– атрибут уведомления **hierarchySelectList**, имеющий в качестве значения цепочку битов, определяющую недействительные опции выбора иерархии.

3) Если в запросе имеются опции выбора иерархии, которые не поддерживаются в DSA и которые не учтены в пункте 2), то **notification** должен содержать:

– атрибут уведомления **searchServiceProblem** со значением **id-pr-unavailableHierarchySelect**;

– атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и

– атрибут уведомления **hierarchySelectList**, имеющий в качестве значения цепочку битов, определяющую неподдержанные опции выбора иерархии.

4) Если в запросе имеются опции управления поиском (определенные в 10.2.1), которые не разрешены, или некоторые опции отсутствуют в комбинации компонентов **defaultControls** и **mandatoryControls** правила поиска, то **notification** должен содержать:

– атрибут уведомления **searchServiceProblem** со значением **id-pr-invalidSearchControlOptions**;

– атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и

– атрибут уведомления **searchControlOptionsList**, имеющий в качестве значения цепочку битов, определяющую недействительные опции управления поиском.

5) Если в запросе имеются опции управления службой, которые не разрешены, или некоторые опции отсутствуют в комбинации компонентов **defaultControls** и **mandatoryControls** правила поиска, то компонент **notification** должен содержать:

– атрибут уведомления **searchServiceProblem** со значением **id-pr-invalidServiceControlOptions**;

– атрибут уведомления **serviceType**, имеющий компонент **serviceType** правила поиска в качестве значения; и

– атрибут уведомления **serviceControlOptionsList**, имеющий в качестве значения цепочку битов, определяющую недействительные опции управления службой.

13.4 Проверка применения сопоставления

В процедуре проверки операции поиска этот подраздел определяет последний шаг проверки, когда считается, что запрос **search** прошел все другие шаги проверки. Правило поиска в случае неудачи на этом последнем шаге помещается в список **MatchProblemSR** (см. п. 19.3.2.2.1 3) Рекомендации МСЭ-Т X.518 | ИСО/МЭК 9594-4).

Если поисковый запрос не выполнил требование **matchingUse**, определенное в п. 16.10.2 Рекомендации МСЭ-Т X.501 | ИСО/МЭК 9594-2, для любого профиля атрибутов запроса, то **notification** для одного профиля атрибутов запроса, который не прошел проверку, должен содержать:

– атрибут уведомления **searchServiceProblem** со значением **id-pr-attributeMatchingViolation**, если условие сопоставления нарушено, или со значением **id-pr-unsupportedMatchingUse**, если правило сопоставления должно применяться неподдерживаемым способом;

– атрибут уведомления **serviceType,** имеющий компонент **serviceType** правила поиска в качестве значения;

– атрибут уведомления **attributeTypeList**, имеющий в качестве единственного значения идентификатор объекта, определяющий тип атрибута; и

– для нарушенного условия сопоставления дополнительные атрибуты уведомления, определенные спецификацией для этого условия соответствия.

ПРИМЕЧАНИЕ. – Если несколько профилей атрибутов запроса не прошли проверку успешно, то выбор профиля, для которого формируется **notification**, является местным вопросом.

Приложение A

Абстрактные службы на языке ASN.1

(Данное Приложение является составной частью настоящей Рекомендации | Международного стандарта)

В этом приложении приведены определения всех типов, значений и информационных объектов ASN.1, содержащихся в этой спецификации Справочника, в виде модуля **DirectoryAbstractService** на языке ASN.1.

```
DirectoryAbstractService {joint-iso-itu-t ds(5) module(1) directoryAbstractService(2) 5}
DEFINITIONS ::=
BEGIN

-- EXPORTS All --
-- Типы и значения, определенные в этом модуле, экспортируются для использования в других модулях ASN.1,
-- содержащихся в спецификациях Справочника, и для использования в других приложениях, которые будут
-- использовать их для доступа к службам Справочника. Другие приложения могут использовать их для
-- своих собственных целей, но это не будет ограничивать расширения и модификации, необходимые для
-- поддержания или улучшения Справочной службы.

IMPORTS

-- из Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2

    attributeCertificateDefinitions, authenticationFramework, basicAccessControl,
    commonProtocolSpecification, directoryShadowAbstractService, distributedOperations,
    enhancedSecurity, id-at, information Framework, selectedAttributeTypes, serviceAdministration,
    upperBounds
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

    Attribute, ATTRIBUTE, AttributeType, AttributeTypeAssertion, AttributeValue,
    AttributeValueAssertion, CONTEXT, ContextAssertion, DistinguishedName,
    MATCHING-RULE, Name, OBJECT-CLASS, RelativeDistinguishedName,
    SupportedAttributes, SupportedContexts
        FROM InformationFramework informationFramework

    RelaxationPolicy
        FROM ServiceAdministration serviceAdministration

    AttributeTypeAndValue
        FROM BasicAccessControl basicAccessControl

      OPTIONALLY-PROTECTED{ }, OPTIONALLY-PROTECTED-SEQ{ }
        FROM EnhancedSecurity enhancedSecurity

-- из Рек. МСЭ-Т X.518 | ИСО/МЭК 9594-4

    AccessPoint, ContinuationReference, Exclusions, OperationProgress, ReferenceType
        FROM DistributedOperations distributedOperations

-- из Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5

    Code, ERROR, id-errcode-abandoned, id-errcode-abandonFailed, id-errcode-attributeError,
    id-errcode-nameError, id-errcode-referral, id-errcode-securityError, id-errcode-serviceError,
    id-errcode-updateError, id-opcode-abandon, id-opcode-addEntry, id-opcode-compare,
    id-opcode-list, id-opcode-modifyDN, id-opcode-modifyEntry, id-opcode-read,
    id-opcode-removeEntry, id-opcode-search, InvokeId, OPERATION
        FROM CommonProtocolSpecification commonProtocolSpecification

-- из Рек. МСЭ-Т X.520 | ИСО/МЭК 9594-6

    DirectoryString {}
        FROM SelectedAttributeTypes selectedAttributeTypes
```

```
        ub-domainLocalID, ub-saslMechanism
            FROM UpperBounds upperBounds
-- из Рек. МСЭ-Т X.509 | ИСО/МЭК 9594-8
        AlgorithmIdentifier, Certification Path, ENCRYPTED {}, SIGNATURE {}, SIGNED {}
            FROM AuthenticationFramework authenticationFramework
        AttributeCertificationPath
            FROM AttributeCertificateDefinitions attributeCertificateDefinitions
-- из Рек. МСЭ-Т X.525 / ИСО/МЭК 9594-9
        AgreementID
            FROM DirectoryShadowAbstractService directoryShadowAbstractService
-- из RFC 2025
        SPKM-ERROR, SPKM-REP-TI, SPKM-REQ
            FROM SpkmGssTokens { iso (1) identified-organization (3) dod(6) internet (1)
                security (5) mechanisms (5) spkm (1) spkmGssTokens (10) }      ;
-- Общие типы данных --
CommonArguments ::= SET {
        serviceControls          [30]  ServiceControls DEFAULT {},
        securityParameters       [29]  SecurityParameters OPTIONAL,
        requestor                [28]  DistinguishedName OPTIONAL,
        operationProgress        [27]  OperationProgress
                                           DEFAULT { nameResolutionPhase notStarted },
        aliasedRDNs              [26]  INTEGER OPTIONAL,
        criticalExtensions       [25]  BIT STRING OPTIONAL,
        referenceType            [24]  ReferenceType OPTIONAL,
        entryOnly                [23]  BOOLEAN DEFAULT TRUE,
        nameResolveOnMaster      [21]  BOOLEAN DEFAULT FALSE,
        operationContexts        [20]  ContextSelection OPTIONAL,
        familyGrouping           [19]  FamilyGrouping DEFAULT entryOnly }

FamilyGrouping ::= ENUMERATED {
        entryOnly       (1),
        compoundEntry   (2),
        strands         (3),
        multiStrand     (4) }

CommonResults ::= SET {
        securityParameters      [30]  SecurityParameters                  OPTIONAL,
        performer               [29]  DistinguishedName                   OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                             DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute OPTIONAL}

        CommonResultsSeq ::= SEQUENCE {
        securityParameters      [30]  SecurityParameters                  OPTIONAL,
        performer               [29]  DistinguishedName                   OPTIONAL,
        aliasDereferenced       [28]  BOOLEAN                             DEFAULT FALSE,
        notification            [27]  SEQUENCE SIZE (1..MAX) OF Attribute OPTIONAL}

ServiceControls ::= SET {
        options                       [0] ServiceControlOptions DEFAULT { },
        priority                      [1]  INTEGER { low (0), medium (1), high (2) } DEFAULT medium,
        timeLimit                     [2]  INTEGER OPTIONAL,
        sizeLimit                     [3]  INTEGER OPTIONAL,
        scopeOfReferral               [4]  INTEGER { dmd(0), country(1) } OPTIONAL,
        attributeSizeLimit            [5]  INTEGER OPTIONAL,
        manageDSAITPlaneRef           [6]  SEQUENCE{
            dsaName                            Name,
            agreementID                        AgreementID } OPTIONAL,
        serviceType                   [7]  OBJECT IDENTIFIER OPTIONAL,
        userClass                     [8]  INTEGER OPTIONAL}
```

```
ServiceControlOptions ::= BIT STRING {
        preferChaining                  (0),
        chainingProhibited              (1),
        localScope                      (2),
        dontUseCopy                     (3),
        dontDereferenceAliases          (4),
        subentries                      (5),
        copyShallDo                     (6),
        partialNameResolution           (7),
        manageDSAIT                     (8),
        noSubtypeMatch                  (9),
        noSubtypeSelection              (10),
        countFamily                     (11),
        dontSelectFriends               (12),
        dontMatchFriends                (13) }

EntryInformationSelection ::= SET {
        attributes                      CHOICE {
                allUserAttributes           [0]  NULL,
                select                      [1]  SET OF AttributeType
                -- пустое множество означает,
                   что не запрашиваются никакие атрибуты -- } DEFAULT allUserAttributes : NULL,
        infoTypes               [2]     INTEGER {
                attributeTypesOnly              (0),
                attributeTypesAndValues         (1) } DEFAULT attributeTypesAndValues,
        extraAttributes                 CHOICE {
                allOperationalAttributes    [3]     NULL,
                select                      [4]     SET SIZE (1..MAX) OF AttributeType} OPTIONAL,
        contextSelection                ContextSelection OPTIONAL,
        returnContexts                  BOOLEAN DEFAULT FALSE,
        familyReturn                    FamilyReturn DEFAULT
                                        { memberSelect contributingEntriesOnly }

ContextSelection ::= CHOICE {
        allContexts             NULL,
        selectedContexts        SET SIZE (1 ..MAX) OF TypeAndContextAssertion}

TypeAndContextAssertion ::= SEQUENCE {
        type                    AttributeType,
        contextAssertions       CHOICE {
                preference              SEQUENCE OF ContextAssertion,
                all                     SET OF ContextAssertion } }

FamilyReturn ::= SEQUENCE {
        memberSelect    ENUMERATED {
            contributingEntriesOnly     (1),
            participatingEntriesOnly    (2),
            compoundEntry               (3) },
        familySelect    SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id OPTIONAL}

EntryInformation ::= SEQUENCE {
        name                            Name,
        fromEntry                       BOOLEAN DEFAULT TRUE,
        information                     SET SIZE (1..MAX) OF CHOICE {
                attributeType                   AttributeType,
                attribute                       Attribute } OPTIONAL,
        incompleteEntry         [3]     BOOLEAN DEFAULT FALSE, -- отсутствует в системах первого издания
        partialName             [4]     BOOLEAN DEFAULT FALSE, -- отсутствует в системах второго и
                                                                  последующего изданий
        derivedEntry            [5]     BOOLEAN DEFAULT FALSE -- отсутствует в системах до четвертого издания -- }

family-information ATTRIBUTE ::= {
        WITH SYNTAX             FamilyEntries
        USAGE                   directoryOperation
        ID                      id-at-family-information }

FamilyEntries ::= SEQUENCE {
        family-class            OBJECT-CLASS.&id,  -- значение структурного класса объекта
        familyEntries           SEQUENCE OF FamilyEntry }
```

```
FamilyEntry ::= SEQUENCE {
        rdn                     RelativeDistinguishedName,
        information             SEQUENCE OF CHOICE {
                attributeType           AttributeType,
                attribute               Attribute },
        family-info             SEQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL}

Filter ::= CHOICE {
        item    [0]     FilterItem,
        and     [1]     SET OF Filter,
        or      [2]     SET OF Filter,
        not     [3]     Filter }

FilterItem ::= CHOICE {
        equality                [0]     AttributeValueAssertion,
        substrings              [1]     SEQUENCE {
                type                        ATTRIBUTE.&id ({ SupportedAttributes }),
                strings                     SEQUENCE OF CHOICE {
                        initial                 [0]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        any                     [1]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        final                   [2]     ATTRIBUTE.&Type
                                                        ({SupportedAttributes}{@substrings.type}),
                        control                         Attribute }}, -- Используется для спецификации следующих пунктов
        greaterOrEqual          [2]     AttributeValueAssertion,
        lessOrEqual             [3]     AttributeValueAssertion,
        present                 [4]     AttributeType,
        approximateMatch        [5]     AttributeValueAssertion,
        extensibleMatch         [6]     MatchingRuleAssertion,
        contextPresent          [7]     AttributeTypeAssertion }

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule            [1]     SET SIZE (1..MAX) OF MATCHING-RULE.&id,
        type                    [2]     AttributeType OPTIONAL,
        matchValue              [3]     MATCHING-RULE.&AssertionType (CONSTRAINED BY{
                        -- matchValue должно быть значением атрибута, указанного полем &AssertionType одного
                        -- из информационных объектов MATCHING-RULE, указанных посредством matchingRule -- } ),
        dnAttributes            [4]     BOOLEAN DEFAULT FALSE}

PagedResultsRequest  ::=  CHOICE  {
        newRequest                      SEQUENCE {
                pageSize                        INTEGER,
                sortKeys                        SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
                reverse                 [1]     BOOLEAN DEFAULT FALSE,
                unmerged                [2]     BOOLEAN DEFAULT FALSE,
                pageNumber              [3]     INTEGER OPTIONAL },
        queryReference                  OCTET STRING,
        abandonQuery            [0]     OCTET STRING

SortKey ::= SEQUENCE {
        type            AttributeType,
        orderingRule    MATCHING-RULE.&id OPTIONAL    }

SecurityParameters ::= SET {
        certification-path              [0]     CertificationPath               OPTIONAL,
        name                            [1]     DistinguishedName               OPTIONAL,
        time                            [2]     Time                            OPTIONAL,
        random                          [3]     BIT STRING                      OPTIONAL,
        target                          [4]     ProtectionRequest               OPTIONAL,
        response                        [5]     BIT STRING                      OPTIONAL,
        operationCode                   [6]     Code                            OPTIONAL,
        attributeCertificationPath      [7]     AttributeCertificationPath      OPTIONAL,
        errorProtection                 [8]     ErrorProtectionRequest          OPTIONAL,
        errorCode                       [9]     Code                            OPTIONAL}

ProtectionRequest ::= INTEGER { none (0), signed (1) }
```

```
Time ::= CHOICE {
        utcTime             UTCTime,
        generalizedTime     GeneralizedTime }

ErrorProtectionRequest ::= INTEGER { none (0), signed (1) }

-- Операции Привязывание и Отвязывание --

directoryBind OPERATION  ::=  {
        ARGUMENT      DirectoryBindArgument
        RESULT        DirectoryBindResult
        ERRORS        { directoryBindError } }

DirectoryBindArgument  ::=  SET {
        credentials       [0]     Credentials OPTIONAL,
        versions          [1]     Versions DEFAULT {v1} }

Credentials  ::=  CHOICE {
        simple            [0]     SimpleCredentials,
        strong            [1]     StrongCredentials,
        external-
        Procedure         [2]     EXTERNAL,
        spkm              [3]     SpkmCredentials,
        sasl              [4]     SaslCredentials }

SimpleCredentials ::= SEQUENCE {
        name              [0]     DistinguishedName,
        validity          [1]     SET {
            time1                 [0]     CHOICE {
                utc                       UTCTime,
                gt                        GeneralizedTime } OPTIONAL,
            time2                 [1]     CHOICE {
                utc                       UTCTime,
                gt                        GeneralizedTime } OPTIONAL,
            random1               [2]     BIT STRING OPTIONAL,
            random2               [3]     BIT STRING OPTIONAL}, OPTIONAL,
        password          [2]             CHOICE {
            unprotected                   OCTET STRING,
            protected                     SIGNATURE {OCTET STRING} } OPTIONAL }

StrongCredentials ::= SET {
        certification-path      [0]     CertificationPath OPTIONAL,
        bind-token              [1]     Token,
        name                    [2]     DistinguishedName OPTIONAL,
        attributeCertification-
        Path                    [3]     AttributeCertificationPath OPTIONAL }

SpkmCredentials ::= CHOICE {
        req     [0]     SPKM-REQ,
        rep     [1]     SPKM-REP-TI }

SaslCredentials  ::=  SEQUENCE {
        mechanism               [0]     DirectoryString { ub-saslMechanism },
        credentials             [1]     OCTET STRING OPTIONAL,
        saslAbort               [2]     BOOLEAN DEFAULT FALSE }

Token ::= SIGNED { SEQUENCE {
        algorithm               [0]     AlgorithmIdentifier,
        name                    [1]     DistinguishedName,
        time                    [2]     UTCTime,
        random                  [3]     BIT STRING,
        response                [4]     BIT STRING OPTIONAL,
        bindIntAlgorithm        [5]     SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindIntKeyInfo          [6]     BindKeyInfo OPTIONAL,
        bindConfAlgorithm       [7]     SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindConfKeyInfo         [8]     BindKeyInfo OPTIONAL

Versions ::= BIT STRING {v1(0), v2(1) }

DirectoryBindResult ::= DirectoryBindArgument
```

```
directoryBindError ERROR ::= {
        PARAMETER               OPTIONALLY-PROTECTED {
                SET {
                        versions            [0]     Versions DEFAULT {v1},
                        error                       CHOICE {
                                serviceError        [1]     ServiceProblem,
                                securityError       [2]     SecurityProblem } } } }

BindKeyInfo ::= ENCRYPTED {BIT STRING }

-- Операции, аргументы и результаты --

read OPERATION ::= {
        ARGUMENT        ReadArgument
        RESULT          ReadResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-read }

ReadArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                selection               [1]     EntryInformationSelection DEFAULT { },
                modifyRightsRequest     [2]     BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                   CommonArguments } }

ReadResult ::= OPTIONALLY-PROTECTED {
        SET {
                entry               [0] EntryInformation,
                modifyRights        [1] ModifyRights OPTIONAL,
                COMPONENTS OF       CommonResults } }

ModifyRights ::=  SET OF SEQUENCE {
        item                        CHOICE {
                entry                   [0]     NULL,
                attribute               [1]     AttributeType,
                value                   [2]     AttributeValueAssertion },
        permission      [3]     BIT STRING {add (0), remove (1), rename (2), move (3) } }

compare OPERATION  ::= {
        ARGUMENT        CompareArgument
        RESULT          CompareResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-compare }

CompareArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                purported               [1]     AttributeValueAssertion,
                COMPONENTS OF                   CommonArguments } }

CompareResult ::= OPTIONALLY-PROTECTED {
        SET {
                name                            Name OPTIONAL,
                matched                 [0]     BOOLEAN,
                fromEntry               [1]     BOOLEAN DEFAULT TRUE,
                matchedSubtype          [2]     AttributeType OPTIONAL,
                COMPONENTS OF                   CommonResults } }

abandon OPERATION ::= {
        ARGUMENT        AbandonArgument
        RESULT          AbandonResult
        ERRORS          { abandonFailed }
        CODE            id-opcode-abandon }

AbandonArgument ::= OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
                invokeID        [0]     InvokeId } }
```

```
AbandonResult ::= CHOICE {
        null                NULL,
        information         OPTIONALLY-PROTECTED-SEQ {
                                SEQUENCE{
                                    invokeID                InvokeId,
                                    COMPONENTS OF           CommonResultsSeq } } }
list    OPERATION  ::=  {
           ARGUMENT     ListArgument
           RESULT       ListResult
           ERRORS       { nameError | serviceError | referral | abandoned | securityError }
           CODE         id-opcode-list }
ListArgument ::= OPTIONALLY-PROTECTED {
        SET {
            object                  [0]     Name,
            pagedResults            [1]     PagedResultsRequest OPTIONAL,
            listFamily              [2]     BOOLEAN DEFAULT FALSE,
            COMPONENTS OF                   CommonArguments } }
ListResult ::= OPTIONALLY-PROTECTED {
        CHOICE {
            listInfo                        SET {
                name                                    Name OPTIONAL,
                subordinates                [1]         SET OF SEQUENCE {
                    rdn                                 RelativeDistinguishedName,
                    aliasEntry                          [0]     BOOLEAN DEFAULT FALSE,
                    fromEntry                           [1]     BOOLEAN DEFAULT TRUE},
                partialOutcomeQualifier     [2]         PartialOutcomeQualifier OPTIONAL,
                COMPONENTS OF                           CommonResults },
            uncorrelatedListInfo    [0]     SET OF ListResult } }
PartialOutcomeQualifier ::= SET {
        limitProblem                    [0]     LimitProblem OPTIONAL,
        unexplored                      [1]     SET SIZE (1..MAX) OF ContinuationReference OPTIONAL,
        unavailableCriticalExtensions   [2]     BOOLEAN DEFAULT FALSE,
        unknownErrors                   [3]     SET SIZE (1..MAX) OF ABSTRACT-SYNTAX.&Type OPTIONAL,
        queryReference                  [4]     OCTET STRING OPTIONAL,
        overspecFilter                  [5]     Filter OPTIONAL,
        notification                    [6]     SEQUENCE SIZE (1 .. MAX) OF Attribute OPTIONAL,
        entryCount                              CHOICE {
            bestEstimate                        [7] INTEGER,
            lowEstimate                         [8] INTEGER,
            exact                               [9]     INTEGER } OPTIONAL,
        streamedResult                  [10]    BOOLEAN DEFAULT  FALSE }
LimitProblem ::= INTEGER {
        timeLimitExceeded (0), sizeLimitExceeded (1), administrativeLimitExceeded (2) }
search OPERATION ::= {
        ARGUMENT        SearchArgument
        RESULT          SearchResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-search }
SearchArgument ::= OPTIONALLY-PROTECTED {
        SET {
            baseObject              [0]     Name,
            subset                  [1]     INTEGER {
              baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
            filter                  [2]     Filter DEFAULT and: { },
            searchAliases           [3]     BOOLEAN DEFAULT TRUE,
            selection               [4]     EntryInformationSelection DEFAULT { },
            pagedResults            [5]     PagedResultsRequest OPTIONAL,
            matchedValuesOnly       [6]     BOOLEAN DEFAULT FALSE,
            extendedFilter          [7]     Filter OPTIONAL,
            checkOverspecified      [8]     BOOLEAN DEFAULT FALSE,
            relaxation              [9]     RelaxationPolicy        OPTIONAL,
            extendedArea            [10]    INTEGER                 OPTIONAL,
```

```
        hierarchySelections     [11]  HierarchySelections     DEFAULT {self},
        searchControlOptions    [12]  SearchControlOptions    DEFAULT { searchAliases },
        joinArguments           [13]  SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
        joinType                [14]  ENUMERATED {
          innerJoin(0), leftOuterJoin(1), fullOuterJoin(2)} DEFAULT leftOuterJoin,
        COMPONENTS OF                 CommonArguments } }

HierarchySelections ::= BIT STRING {
    self            (0),
    children        (1),
    parent          (2),
    hierarchy       (3),
    top             (4),
    subtree         (5),
    siblings        (6),
    siblingChildren (7),
    siblingSubtree  (8),
    all             (9) }

SearchControlOptions ::= BIT STRING {
    searchAliases           (0),
    matchedValuesOnly       (1),
    checkOverspecified      (2),
    performExactly          (3),
    includeAllAreas         (4),
    noSystemRelaxation      (5),
    dnAttribute             (6),
    matchOnResidualName     (7),
    entryCount              (8),
    useSubset               (9),
    separateFamilyMembers   (10),
    searchFamily            (11) }

JoinArgument ::= SEQUENCE {
    joinBaseObject      [0]   Name,
    domainLocalID       [1]   DomainLocalID OPTIONAL,
    joinSubset          [2]   ENUMERATED {
                baseObject(0), oneLevel(1), wholeSubtree(2) } DEFAULT baseObject,
    joinFilter          [3]   Filter OPTIONAL,
    joinAttributes      [4]   SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
    joinSelection       [5]   EntryInformationSelection }

DomainLocalID  :=  DirectoryString { ub-domainLocalID }

JoinAttPair  ::=  SEQUENCE {
    baseAtt             AttributeType,
    joinAtt             AttributeType,
    joinContext         SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL}

JoinContextType  ::= CONTEXT.&id({SupportedContexts})

SearchResult ::= OPTIONALLY-PROTECTED {
    CHOICE {
        searchInfo                  SET {
        name                                Name OPTIONAL,
        entries                     [0]     SET OF EntryInformation,
        partialOutcomeQualifier     [2]     PartialOutcomeQualifier OPTIONAL,
        altMatching                 [3] BOOLEAN DEFAULT FALSE,
        COMPONENTS OF                       CommonResults },
    uncorrelatedSearchInfo  [0]   SET OF SearchResult } }

addEntry OPERATION ::= {
    ARGUMENT    AddEntryArgument
    RESULT      AddEntryResult
    ERRORS      { attributeError | nameError | serviceError | referral | securityError |
                updateError }
    CODE        id-opcode-addEntry }
```

```
AddEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                entry                   [1]     SET OF Attribute,
                targetSystem            [2]     AccessPoint OPTIONAL,
                COMPONENTS OF                   CommonArguments} }

AddEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }

removeEntry OPERATION ::= {
        ARGUMENT        RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          { nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-removeEntry }

RemoveEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                COMPONENTS OF                   CommonArguments } }

RemoveEntryResult ::= CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                SEQUENCE { COMPONENTS OF CommonResultsSeq } } }

modifyEntry OPERATION ::= {
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          { attributeError | nameError | serviceError | referral | securityError |
                        updateError }
        CODE            id-opcode-modifyEntry }

ModifyEntryArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     Name,
                changes                 [1]     SEQUENCE OF EntryModification,
                selection               [2]     EntryInformationSelection OPTIONAL,
                COMPONENTS OF                   CommonArguments } }

ModifyEntryResult ::= CHOICE {
        null                    NULL,
        information             OPTIONALLY-PROTECTED-SEQ
                SEQUENCE{
                        entry           [0]     EntryInformation OPTIONAL,
                        COMPONENTS OF           CommonResultsSeq } } }

EntryModification::=    CHOICE {
        addAttribute    [0]     Attribute,
        removeAttribute [1]     AttributeType,
        addValues       [2]     Attribute,
        removeValues    [3]     Attribute,
        alterValues     [4]     AttributeTypeAndValue,
        resetValue      [5]     AttributeType
        replaceValues   [6]     Attribute }

modifyDN OPERATION ::= {
        ARGUMENT        ModifyDNArgument
        RESULT          ModifyDNResult
        ERRORS          { nameError | serviceError | referral | securityError | updateError }
        CODE            id-opcode-modifyDN }

ModifyDNArgument ::= OPTIONALLY-PROTECTED {
        SET {
                object                  [0]     DistinguishedName,
                newRDN                  [1]     RelativeDistinguishedName,
                deleteOldRDN            [2]     BOOLEAN DEFAULT FALSE,
                newSuperior             [3]     DistinguishedName OPTIONAL,
```

```
            COMPONENTS OF           CommonArguments } }
ModifyDNResult ::= CHOICE {
        null                NULL,
        information         OPTIONALLY-PROTECTED-SEQ {
            SEQUENCE{
                newRDN              RelativeDistinguishedName,
                COMPONENTS OF       CommonResultsSeq } } }
```

-- Ошибки и параметры --

```
abandoned ERROR ::= {   -- не является фактически "ошибкой"
        PARAMETER   OPTIONALLY-PROTECTED {
            SET {COMPONENTS OF CommonResults} }
        CODE            id-errcode-abandoned }

abandonFailed ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED {
            SET{
                problem             [0]     AbandonProblem,
                operation           [1]     InvokeId,
                COMPONENTS OF               CommonResults } }
        CODE                id-errc    ode-abandonFailed }

AbandonProblem ::= INTEGER { noSuchOperation (1), tooLate (2), cannotAbandon (3) }

attributeError ERROR ::= {
            PARAMETER   OPTIONALLY-PROTECTED
                SET{
                    object          [0]     Name,
                    problems        [1]     SET OF SEQUENCE {
                        problem         [0]     AttributeProblem,
                        type            [1]     AttributeType,
                        value           [2]     AttributeValue OPTIONAL},
                    COMPONENTS OF   CommonResults } }
            CODE            id-errcode-attributeError }

AttributeProblem ::= INTEGER {
        noSuchAttributeOrValue              (1),
        invalidAttributeSyntax              (2),
        undefinedAttributeType              (3),
        inappropriateMatching               (4),
        constraintViolation                 (5),
        attributeOrValueAlreadyExists       (6),
        contextViolation                    (7) }

nameError ERROR ::= {
        PARAMETER   OPTIONALLY-PROTECTED {
            SET{
                problem             [0]     NameProblem,
                matched             [1]     Name,
                COMPONENTS OF               CommonResults } }
        CODE                id-errcode-nameError }

NameProblern ::= INTEGER {
        noSuchObject                        (1),
        aliasProblem                        (2),
        invalidAttributeSyntax              (3),
        aliasDereferencingProblem           (4),
        contextProblem                      (5) }

referral ERROR ::= { -- не является фактически "ошибкой"
        PARAMETER OPTIONALLY-PROTECTED {
            SET {
                candidate           [0]     ContinuationReference,
                COMPONENTS OF               CommonResults } }
        CODE                id-errcode-referral }
```

```
securityError ERROR ::= {
        PARAMETER    OPTIONALLY-PROTECTED {
                SET {
                    problem                 [0]    SecurityProblem,
                    spkmInfo                [1]    SPKM-ERROR,
                    COMPONENTS OF                  CommonResults } }
        CODE                id-errcode-securityError }

SecurityProblem ::= INTEGER {
        inappropriateAuthentication     (1),
        invalidCredentials              (2),
        insufficientAccessRights        (3),
        invalidSignature                (4),
        protectionRequired              (5),
        noInformation                   (6),
        blockedCredentials              (7),
        invalidQOPMatch                 (8),
        spkmError                       (9) }

serviceError ERROR ::= {
        PARAMETER    OPTIONALLY-PROTECTED {
                SET{
                        problem             [0]    ServiceProblem,
                        COMPONENTS OF               CommonResults } }
        CODE                id-errcode-serviceError }

ServiceProblem ::= INTEGER {
        busy                            (1),
        unavailable                     (2),
        unwillingToPerform              (3),
        chainingRequired                (4),
        unableToProceed                 (5),
        invalidReference                (6),
        timeLimitExceeded               (7),
        administrativeLimitExceeded     (8),
        loopDetected                    (9),
        unavailableCriticalExtension    (10),
        outOfScope                      (11),
        ditError                        (12),
        invalidQueryReference           (13),
        requestedServiceNotAvailable    (14),
        unsupportedMatchingUse          (15),
        ambiguousKeyAttributes          (16),
        saslBindInProgress              (17) }

updateError ERROR ::= {
        PARAMETER    OPTIONALLY-PROTECTED {
                SET {
                        problem             [0]    UpdateProblem,
                        attributeInfo       [1]    SET SIZE (1..MAX) OF CHOICE {
                            attributeType                  AttributeType,
                            attribute                      Attribute } OPTIONAL,
                        COMPONENTS OF               CommonResults} }
        CODE                            id-errcode-updateError }

UpdateProblem ::= INTEGER {
        namingViolation                 (1),
        objectClassViolation            (2),
        notAllowedOnNonLeaf             (3),
        notAllowedOnRDN                 (4),
        entryAlreadyExists              (5),
        affectsMultipleDSAs             (6),
```

```
        objectClassModificationProhibited       (7),
        noSuchSuperior                          (8),
        notAncestor                             (9),
        parentNotAncestor                       (10),
        hierarchyRuleViolation                  (11),
        familyRuleViolation                     (12) }

-- Типы атрибутов --

id-at-family-information        OBJECT IDENTIFIER    ::= {id-at 64}

END -- DirectoryAbstractService
```

Приложение B

Семантика операций для Управления базовым доступом

(Данное Приложение не является составной частью настоящей Рекомендации | Международного стандарта)

Это приложение содержит диаграммы, которые описывают семантику, связанную с Управлением базовым доступом в том виде, в котором она применяется для обработки операций Справочника (см. рис. от B.1 до B.16).

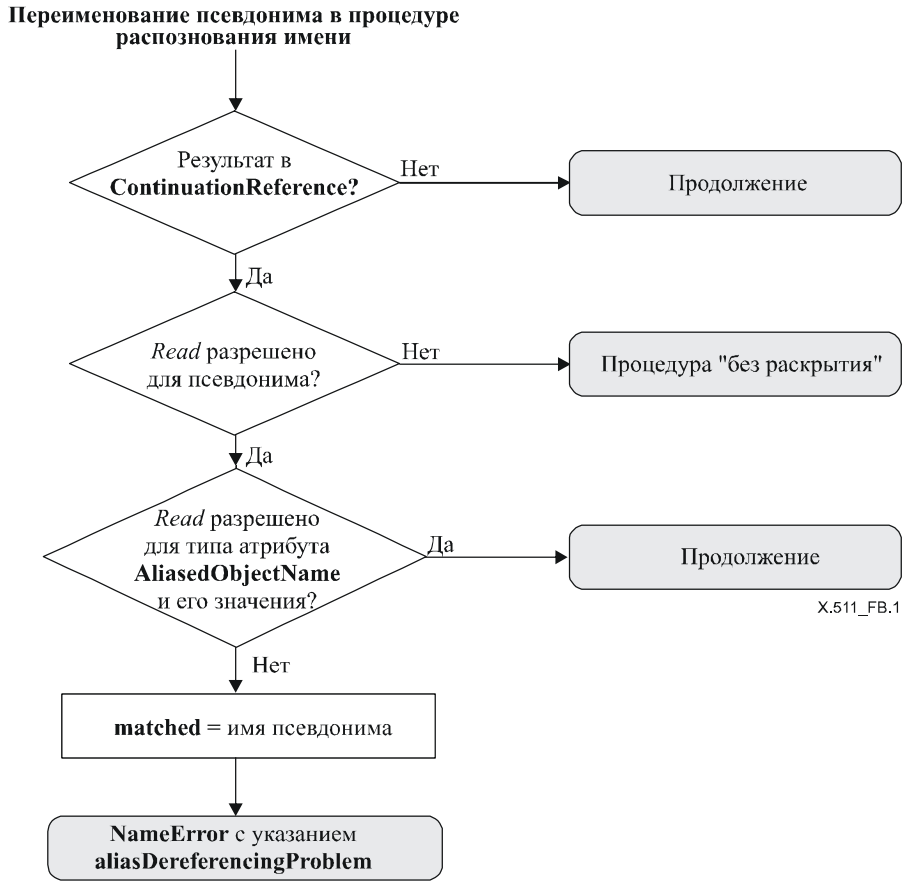

Рисунок B.1 – Переименование псевдонима в процедуре распознавания имени

Рисунок B.2 – Выдача уведомления об ошибке имени

Рисунок B.3 – Процедура без раскрытия существования статьи

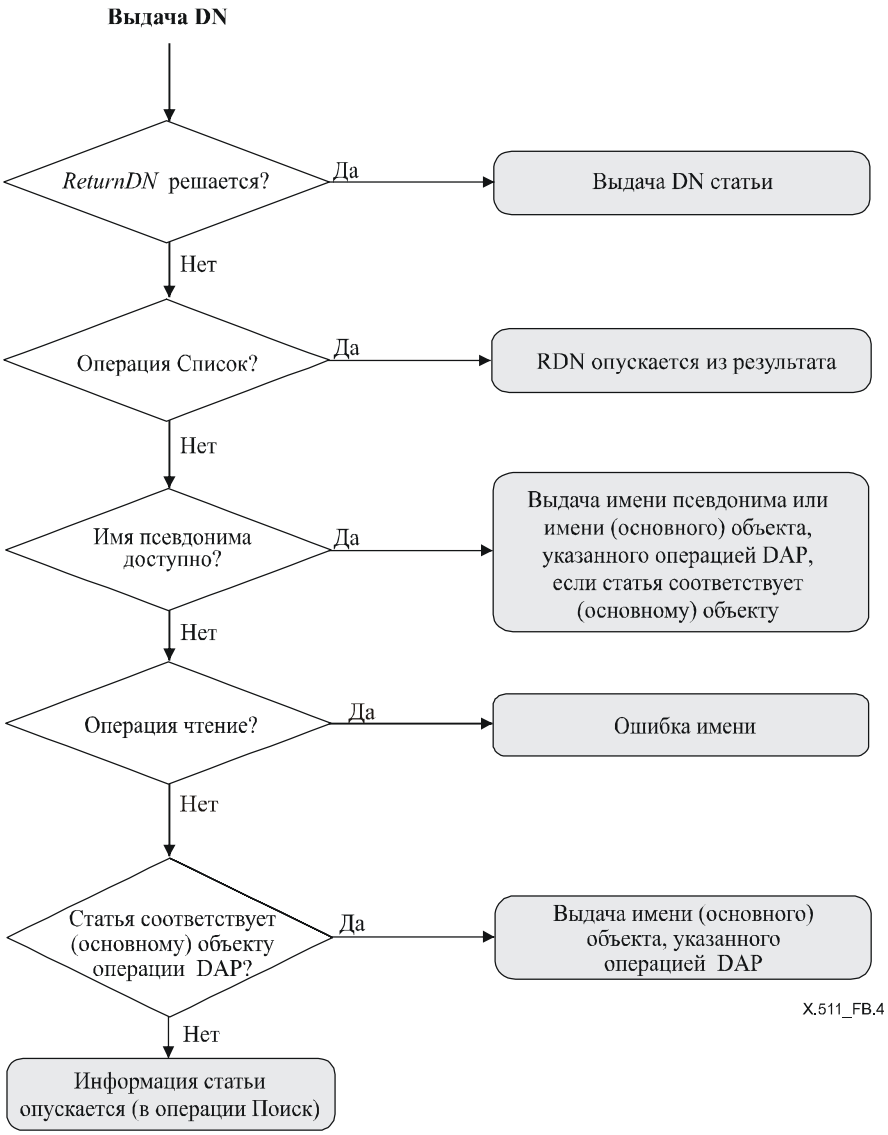

Рисунок B.4 – Выдача Выделенного имени

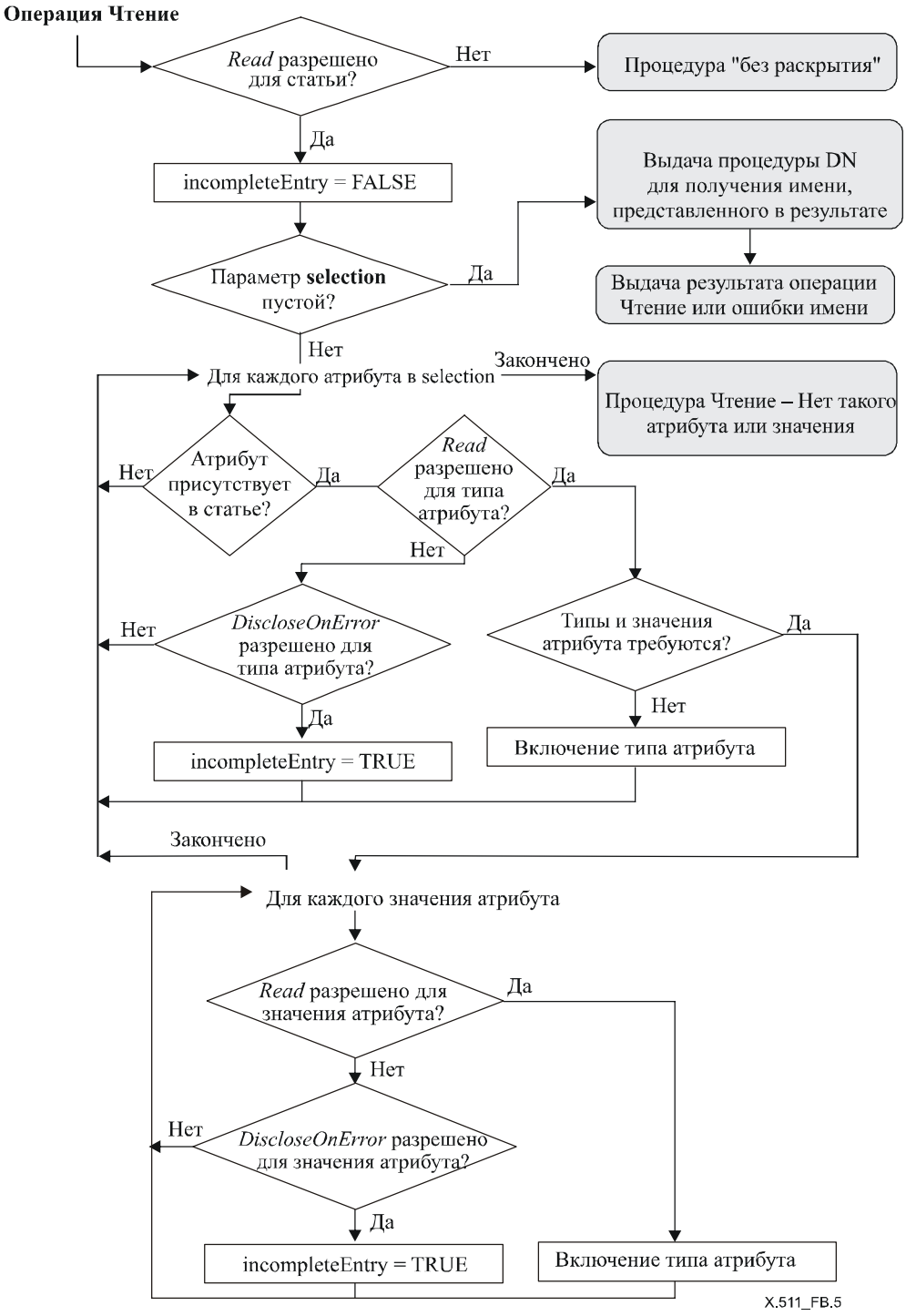

Рисунок B.5 – Операция Чтение

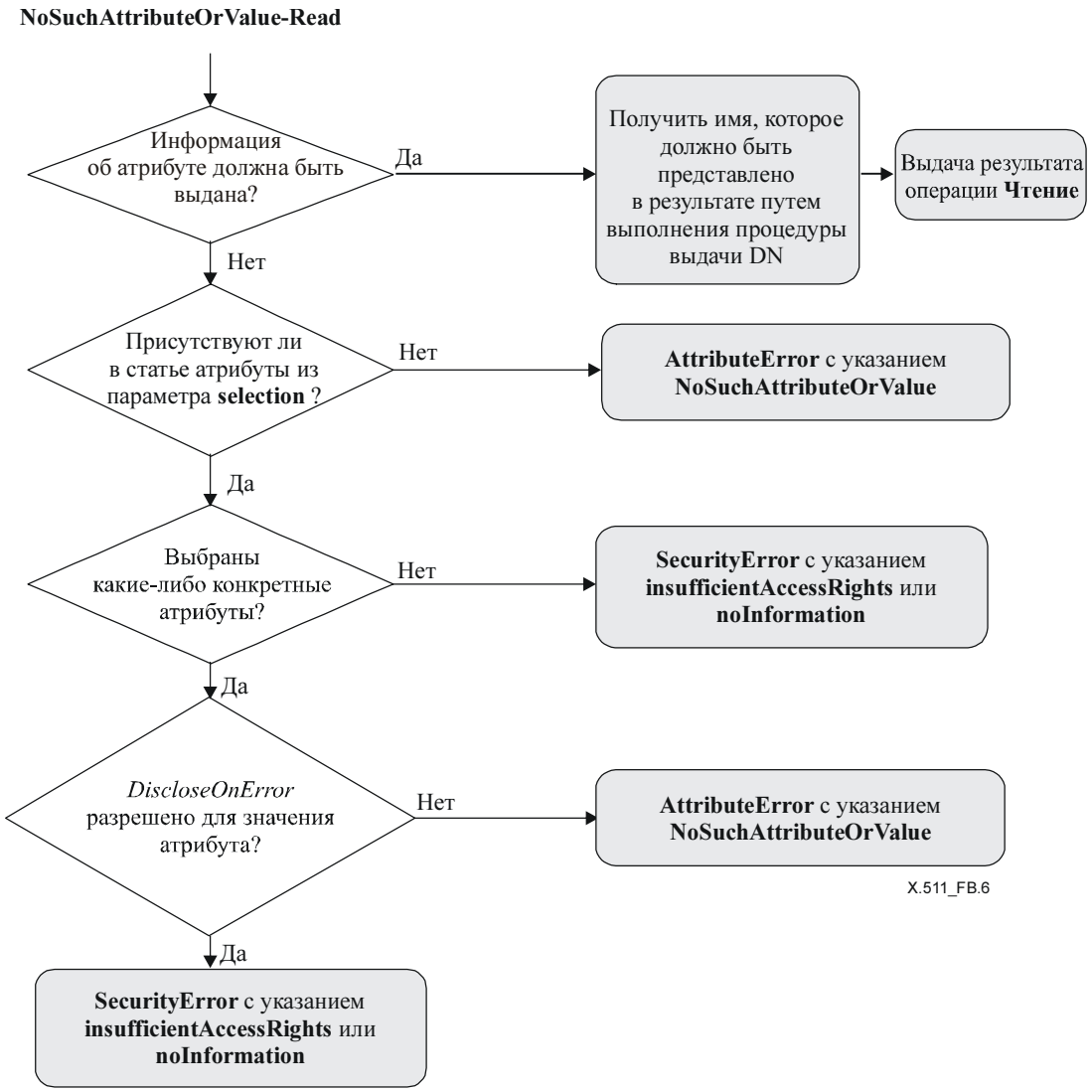

Рисунок B.6 – Для чтения нет такого атрибута или значения

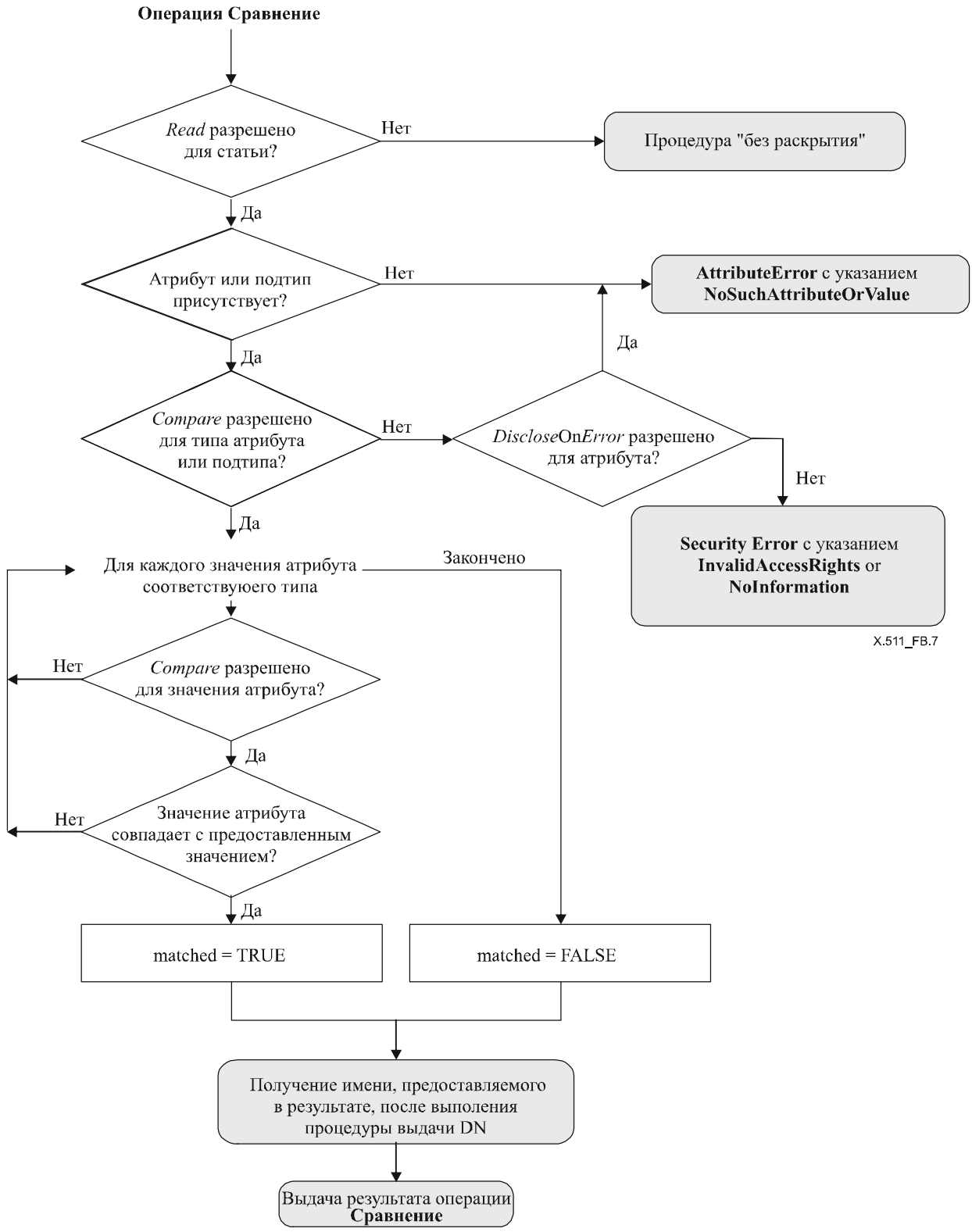

Рисунок B.7 – Операция Сравнение

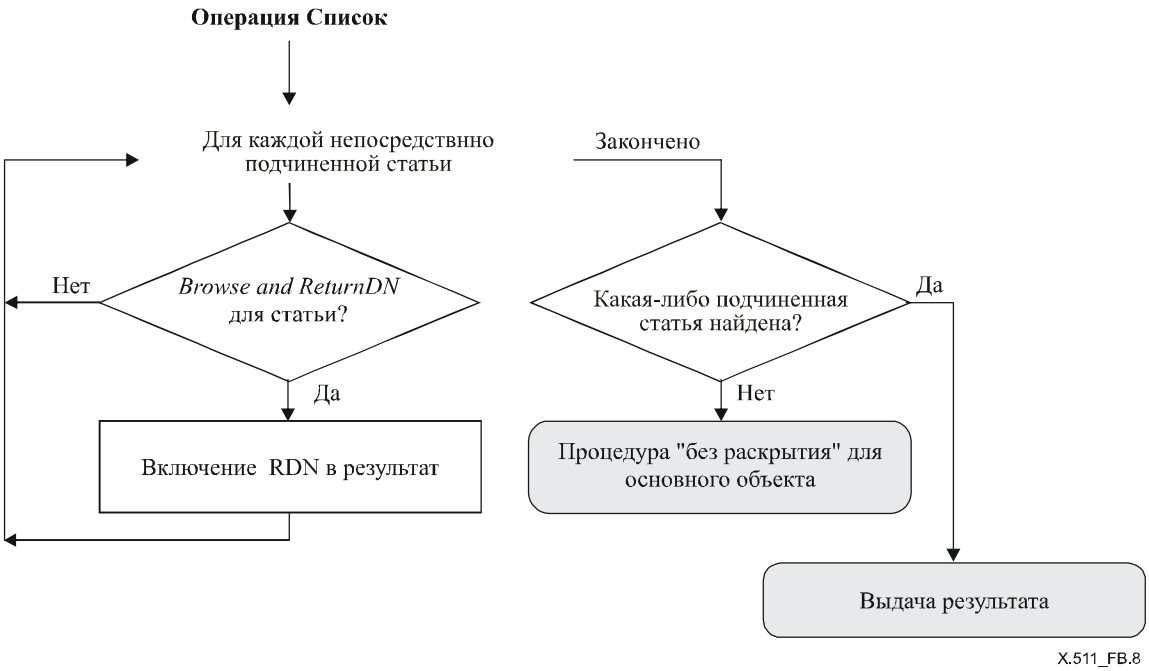

Рисунок B.8 – Операция Список

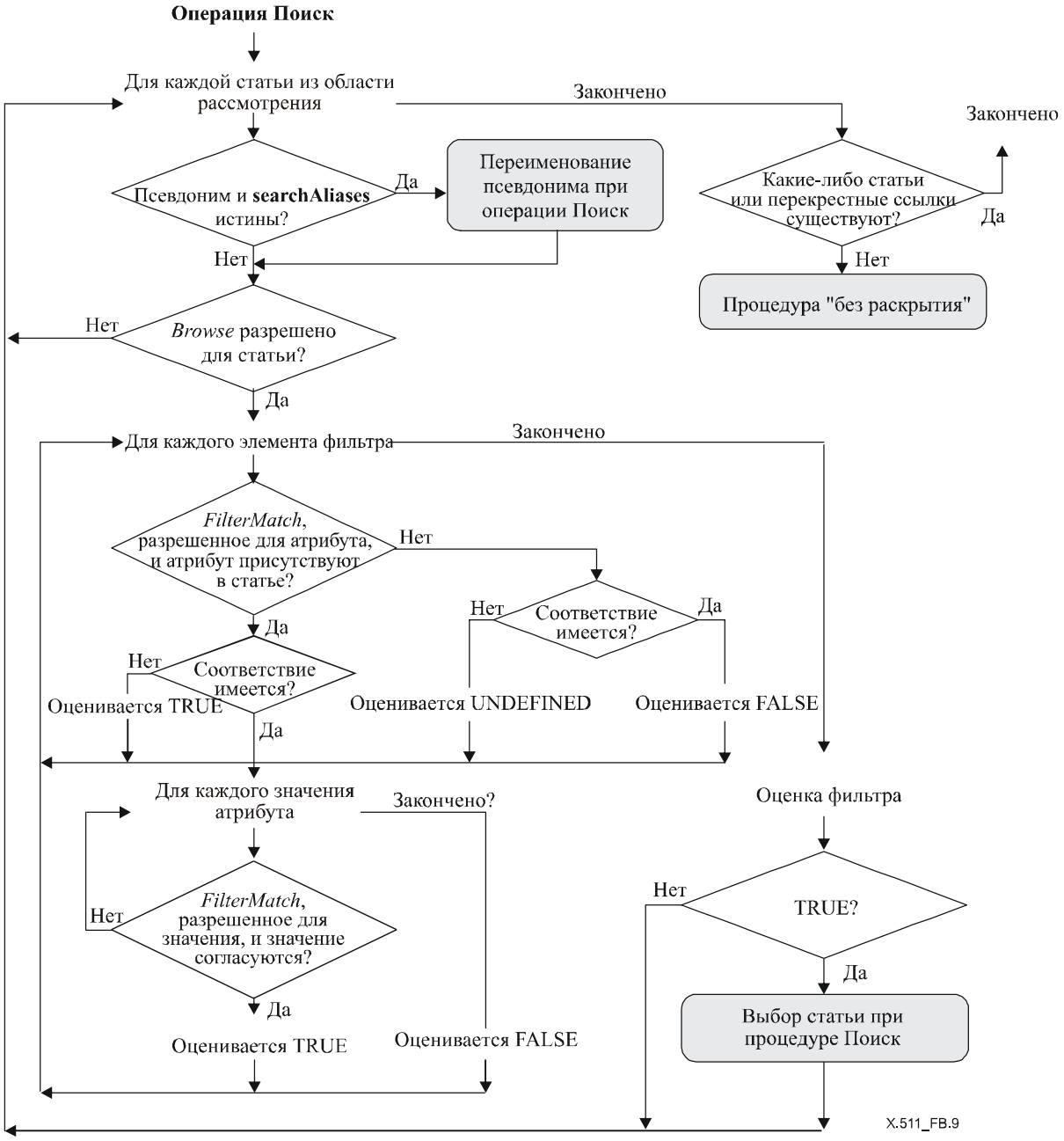

Рисунок B.9 – Операция Поиск

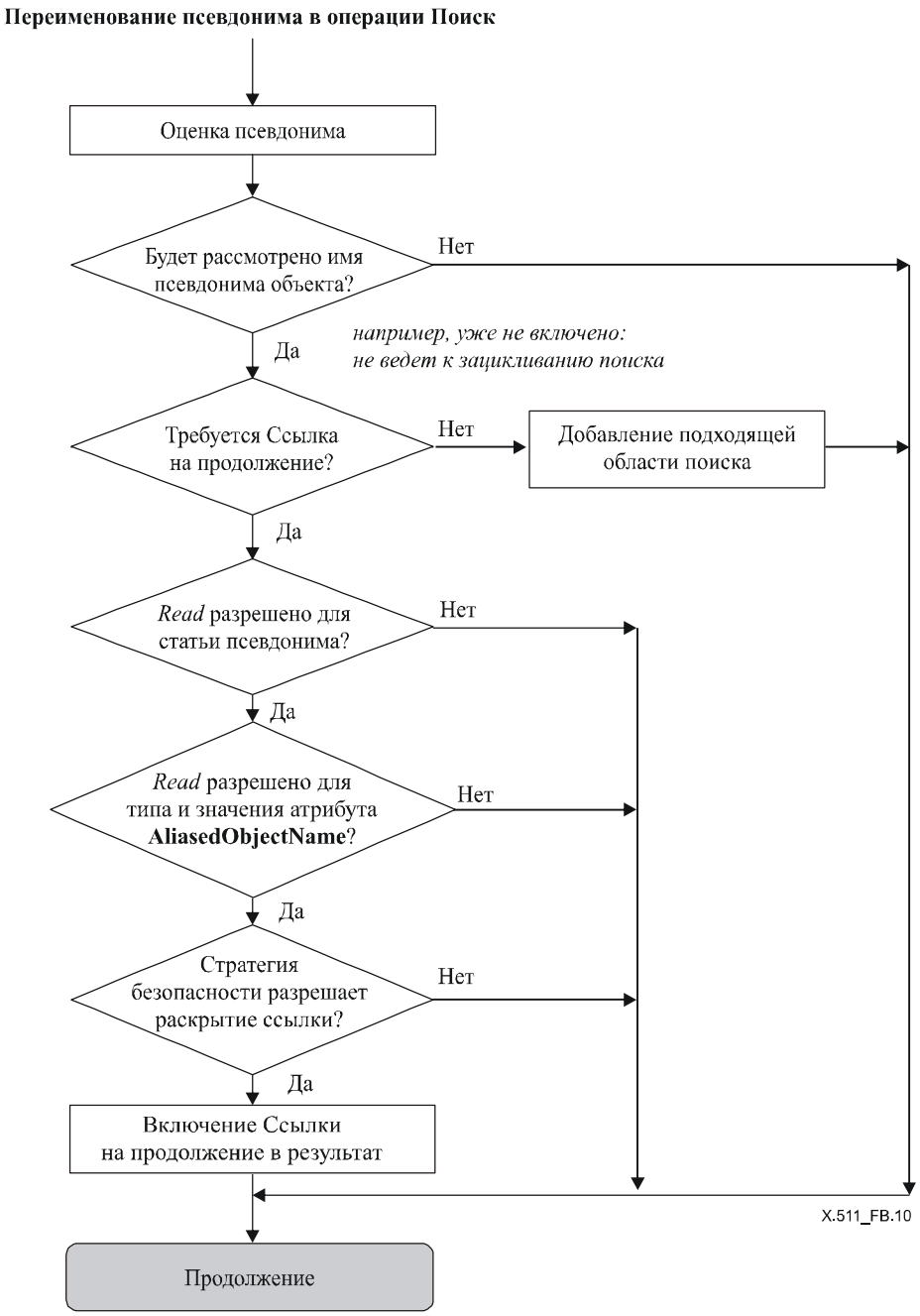

Рисунок B.10 – Переименование псевдонима в операции Поиск

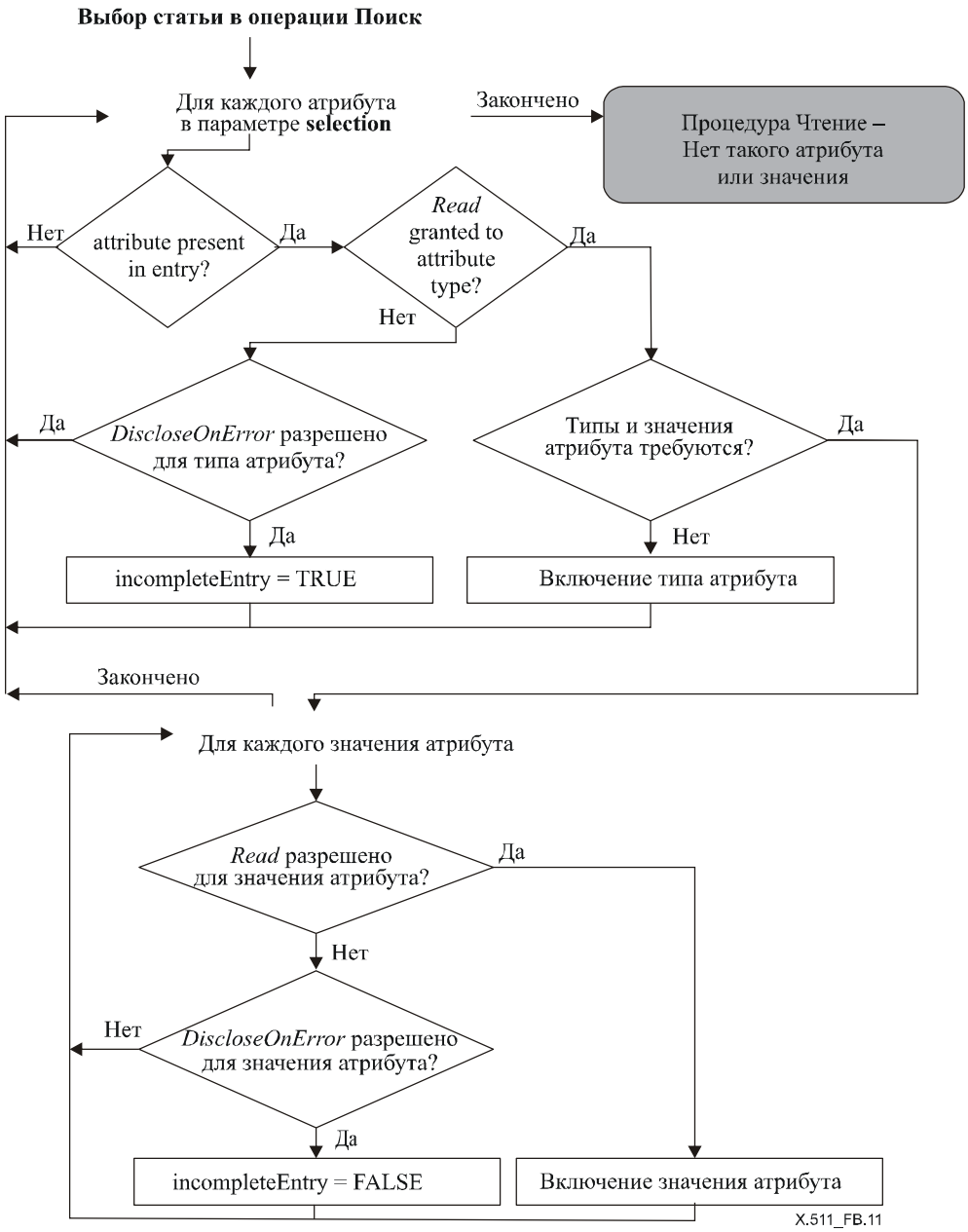

Рисунок B.11 – Выбор статьи в операции Поиск

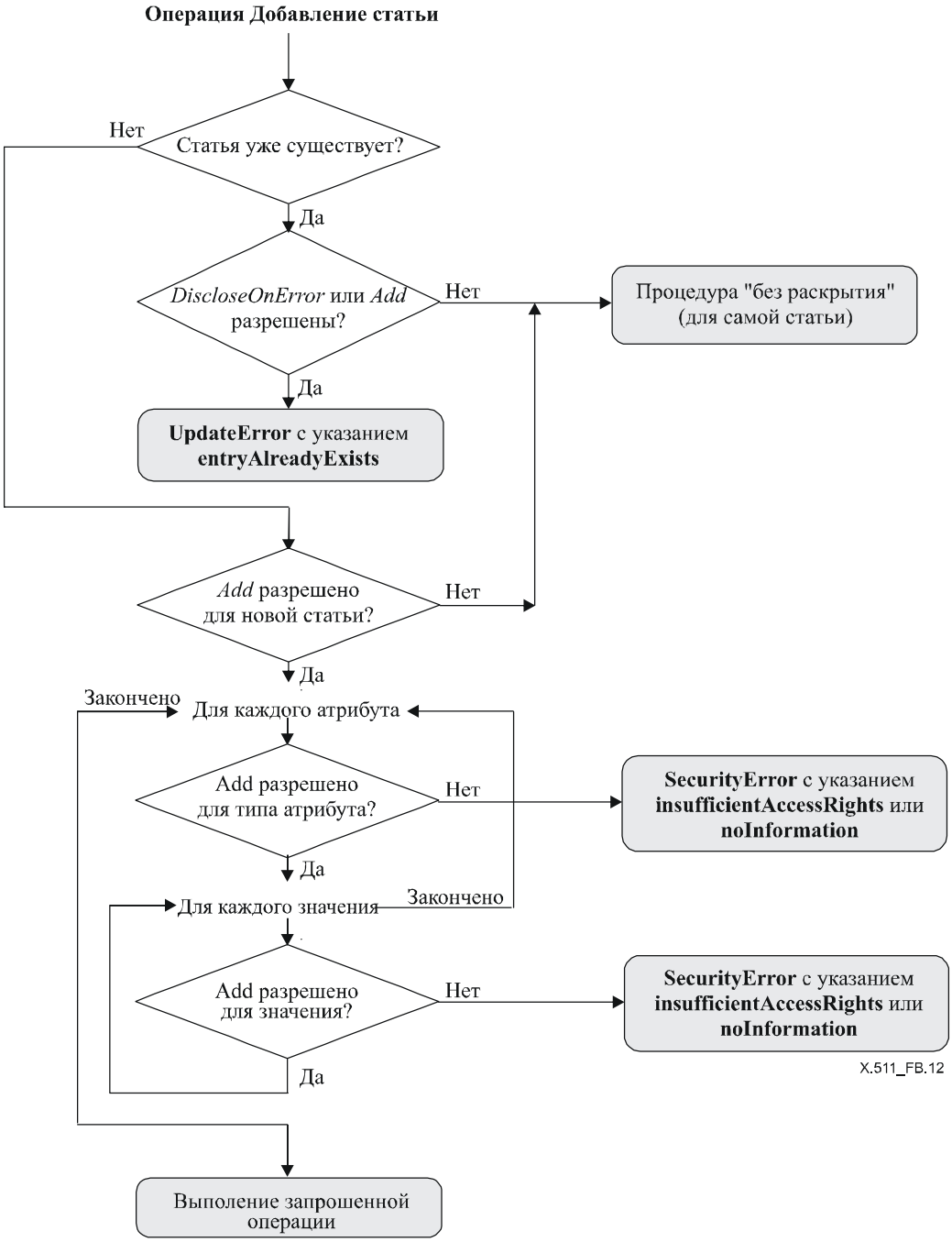

Рисунок B.12 – Операция Добавление Статьи

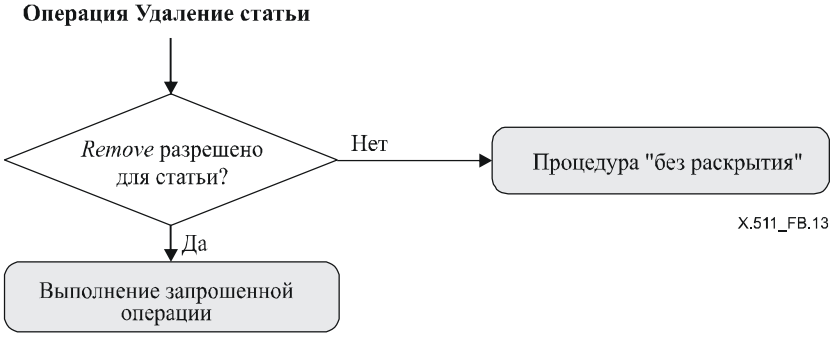

Рисунок B.13 – Операция Удаление статьи

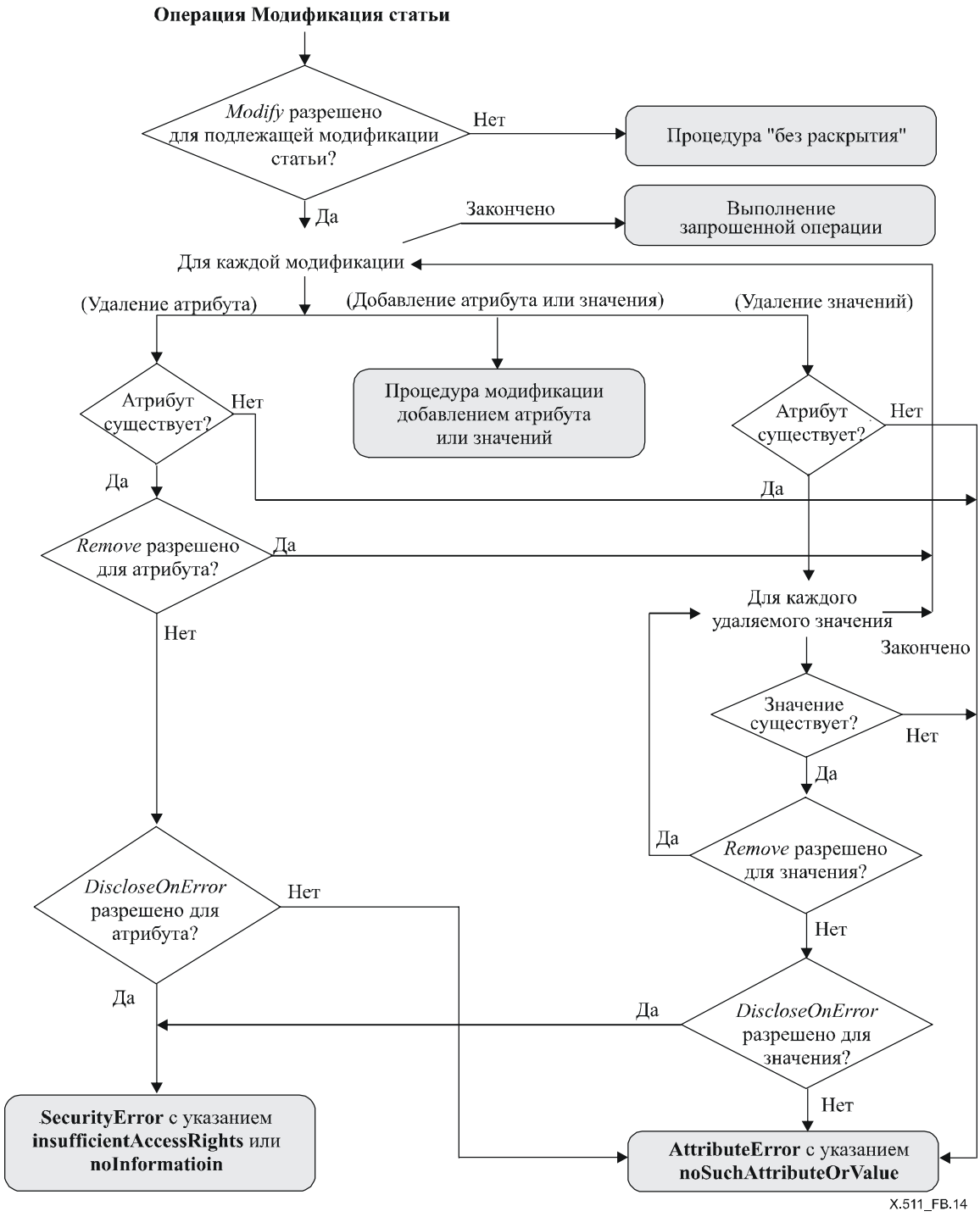

Рисунок B.14 – Операция Модификация статьи

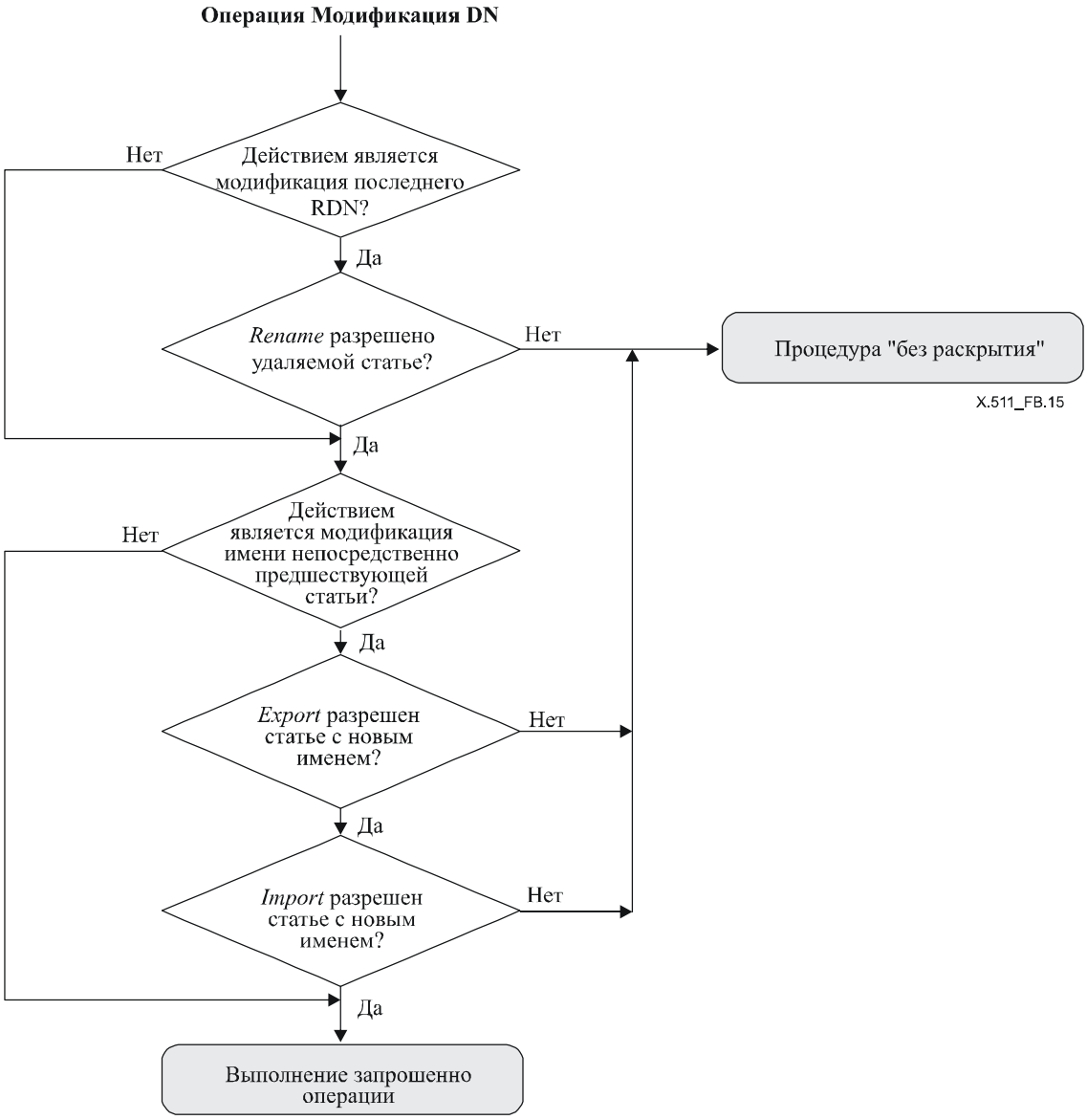

Рисунок B.15 – Операция Модификация DN

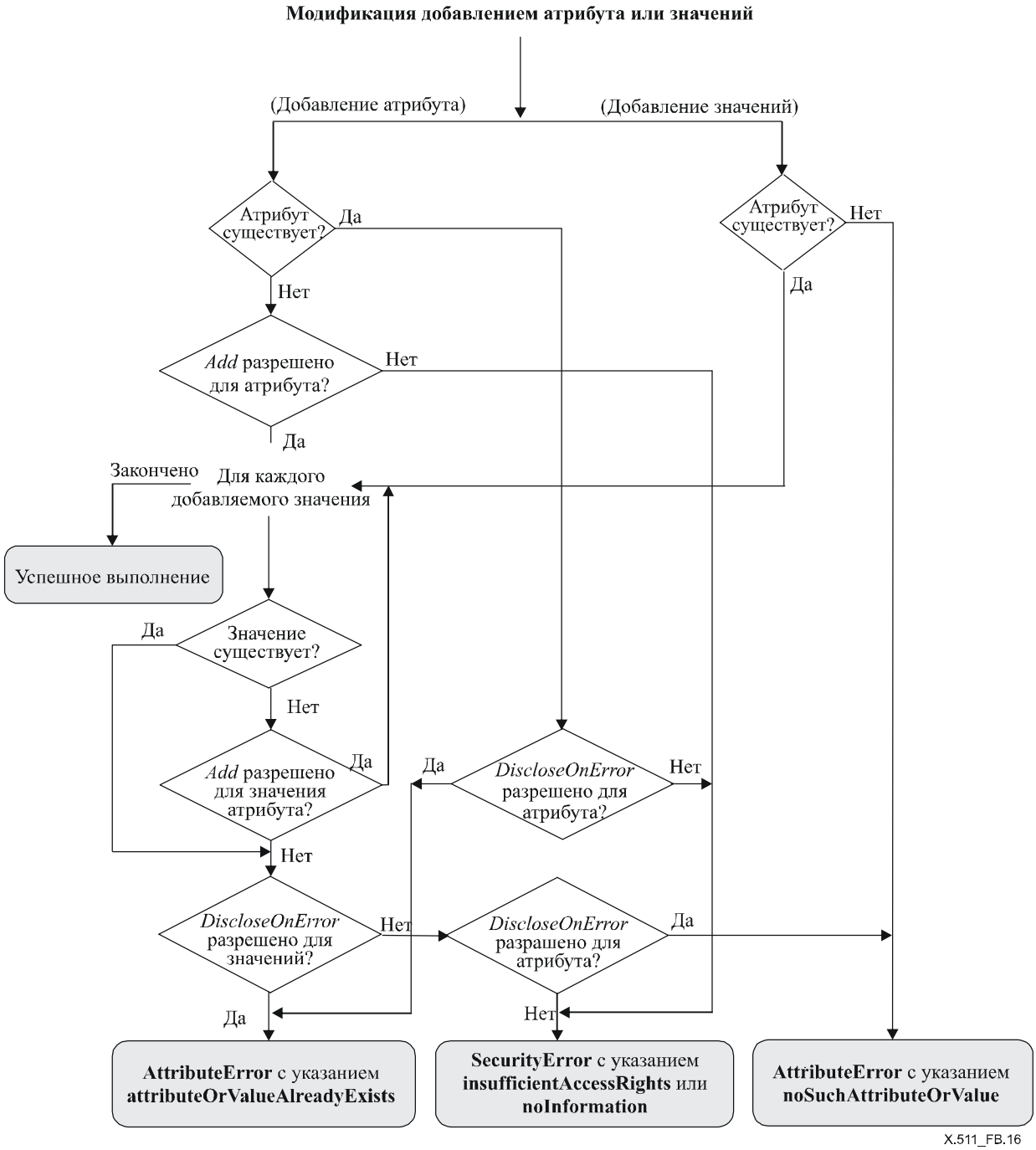

Рисунок B.16 – Модификация добавлением атрибута или значений

Приложение C

Примеры поисковых семейств статей

(Данное Приложение не является составной частью настоящей Рекомендации | Международного стандарта)

C.1 Пример с одиночным семейством

Предположим, что Чарльз Смит (Charles Smith) имеет различные виды связи: стационарный телефон, факсимильный аппарат, мобильный телефон и электронную почту, причем каждый вид имеет свои собственные определяющие параметры. Предположим далее, что Чарльз Смит имеет два счета электронной почты, один на месте его работы и один дома, и что оба предлагают почтовые ящики POP3 (Post Office Protocol 3, почтовый протокол 3) и серверы SMTP (Simple Mail Transfer Protocol, упрощенный протокол электронной почты). Вся эта информация может храниться в виде составной статьи, в которой член Чарльз Смит является порождающим членом, каждый режим связи – порожденным членом, а каждая служба электронной почты – порожденным членом под режимом электронной почты. Это показано ниже на рис. C.1. Поскольку все члены, непосредственно подчиненные порождающему члену, имеют тот же самый структурный класс объекта (**comAddr**), составная статья состоит из одного семейства.

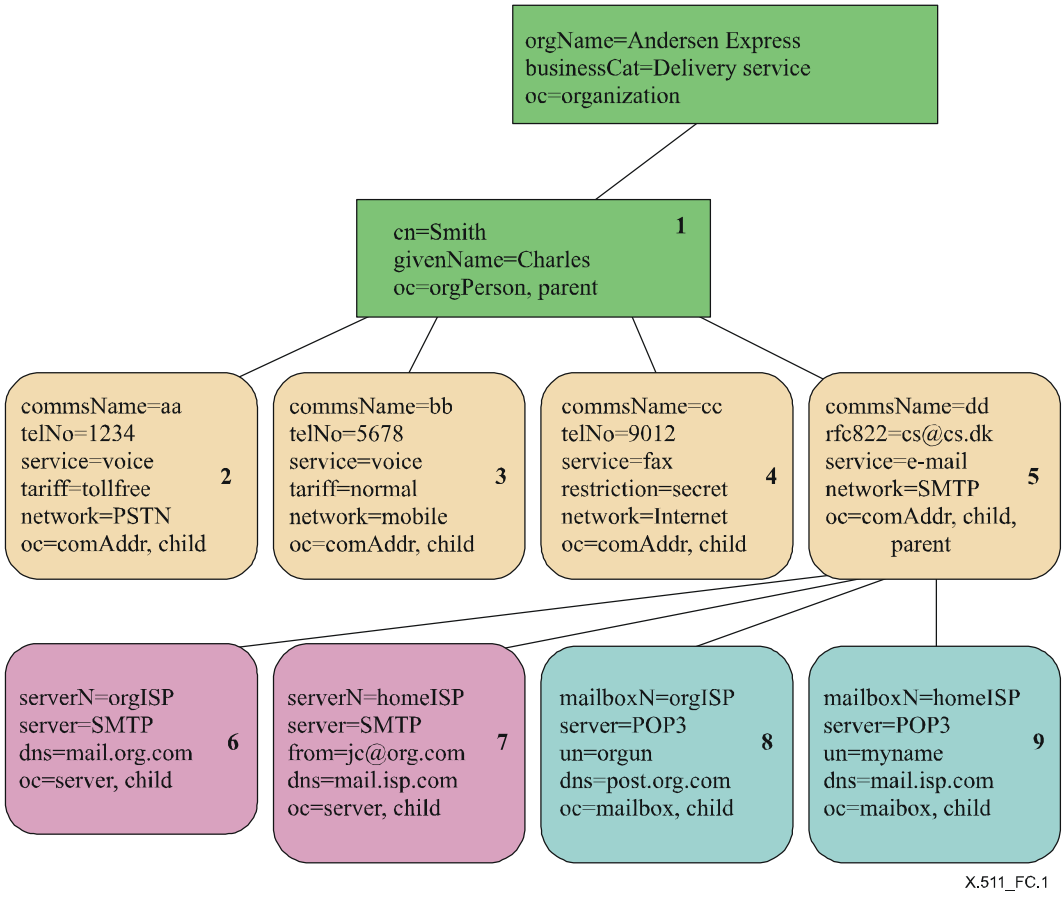

Рисунок C.1 – Семейство статей для случая Чарльза Смита

Предположим, что запрос **search** сгенерирован с основным объектом {... o=Andersen Express}, фильтром {telNo=1234 & tariff=normal} и подмножеством **wholeSubtree** или **oneLevel**. С параметром **familyGrouping**, установленным в следующее состояние:

a) **entryOnly**: Никакие члены семейства не соответствовали бы фильтру.

b) **strands** или **multiStrand**: Никакая трасса или группа трасс в семействе не соответствовала бы фильтру.

c) **compoundEntry**: Член 2 и член 3 вместе соответствовали бы фильтру и были бы отмечены как содействующие члены. Все члены отмечены как участвующие члены.

Ничего из этой составной статьи не будет выдано в случаях a) и b).

Для случая c) выдаваемая информация зависела бы от характеристик семейства в части выдачи (например, от параметра **familyReturn** в **EntryInformationSelection**):

i) **contributingEntriesOnly**: Члены, отмеченные как содействующие элементы, т. е. элементы 2 и 3, будут выданы.

ii) **participatingEntriesOnly** и **compoundEntry**: Все члены из составной статьи будут выданы.

C.2 Пример с несколькими семействами

Предположим, что Чарльз Смит имеет только один стационарный телефон и электронную почту, но также имеет два почтовых адреса с соответствующими параметрами. Вся эта информация может храниться в виде составной статьи, в которой член Чарльз Смит является порождающим членом, а каждый режим связи или почтовый адрес – порожденным членом. Это показано ниже на рис. C.2. Члены, непосредственно подчиненные порождающему члену, входят в два различных структурных класса объектов (**comAddr** и **postAddr**), поэтому составная статья состоит из двух семейств, где члены 1, 2 и 3 составляют одно семейство, а члены 1, 4, 5, 6, 7, 8, и 9 составляют другое семейство.

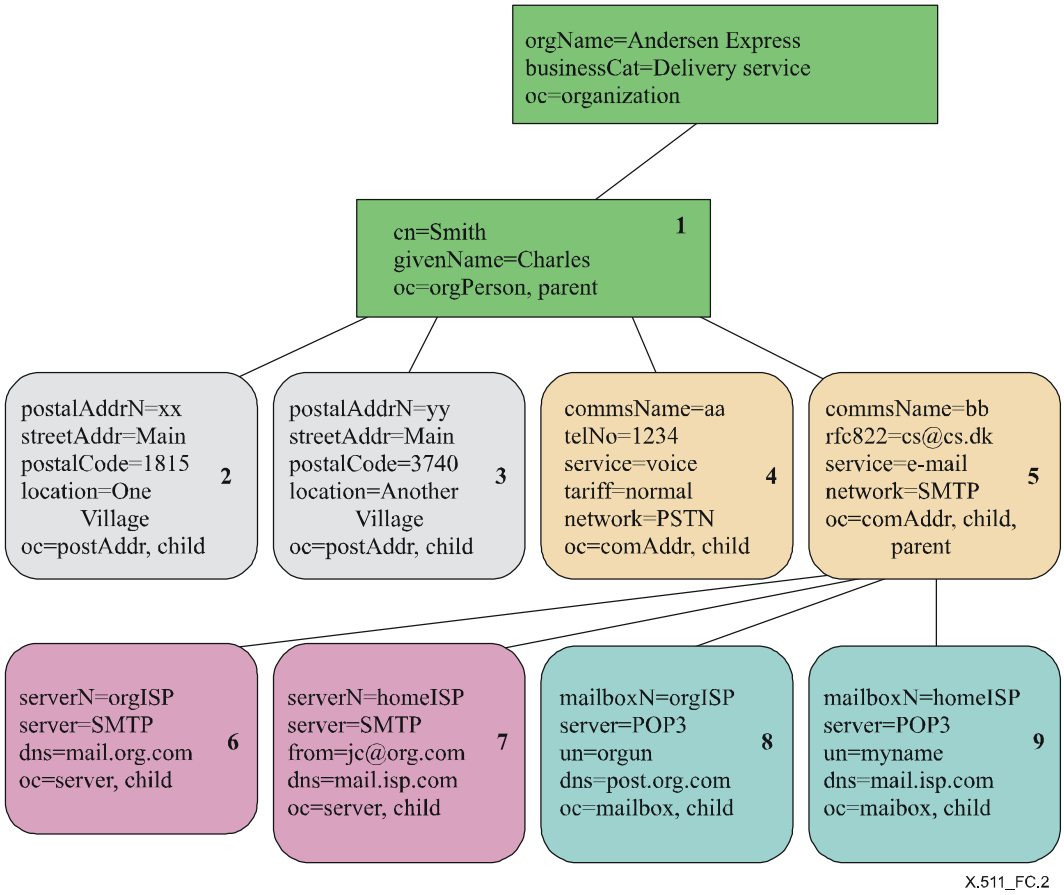

Рисунок C.2 – Два семейства статей для случая Чарльза Смита

C.2.1 Пример фильтра 1

Предположим теперь, что запрос **search** сгенерирован с основным объектом {... o=Andersen Express}, фильтром {telNo=1234 & service=e-mail & streetAddr=Main & postalCode=3740} и подмножеством **wholeSubtree** или **oneLevel**. С параметром **familyGrouping**, установленным в следующее состояние:

a) **entryOnly**: Никакой отдельный член составной статьи не соответствовал бы фильтру.

b) **strands**: Никакая одиночная трасса в каком-либо семействе не соответствовала бы фильтру.

c) **multiStrand**: Никакая комбинация трасс, взятых по одной из каждого семейства, не соответствовала бы фильтру.

d) **compoundEntry**: Члены 2, 3, 4 и 5 вместе соответствовали бы фильтру и были бы отмечены как содействующие члены. Все члены отмечены как участвующие члены.

Ничего из этой составной статьи не будет выдано в случаях a), b) и c).

Для случая d) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Члены, отмеченные как содействующие члены, т. е. члены 2, 3, 4 и 5 будут выданы.

ii) **participatingEntriesOnly** и **compoundEntry**: Все члены из составной статьи будут выданы.

C.2.2 Пример фильтра 2

Теперь изменим фильтр на {rfc822=cs@cs.dk & service=e-mail & streetAddr=Main & postalCode=1815}. С параметром **familyGrouping,** установленным в следующее состояние:

a) **entryOnly**: Никакой отдельный член составной статьи не соответствовал бы фильтру.

b) **strands**: Никакая одиночная трасса в каком-либо семействе не соответствовала бы фильтру.

c) **multiStrand**: Трасса, которая заканчивается на члене 2, вместе с любой трассой, проходящей через член 5, соответствовала бы фильтру. Члены 2 и 5 внесли бы вклад в соответствие и были бы отмечены как содействующие элементы. Члены 1, 2, 5, 6, 7, 8 и 9 отмечены как участвующие элементы.

d) **compoundEntry**: Члены 2 и 5 вместе соответствовали бы фильтру и были бы отмечены как содействующие члены. Все члены отмечены как участвующие члены.

Ничего из этой составной статьи не будет выдано в случаях a) и b).

Для случая c) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Члены, отмеченные как содействующие члены, т. е. члены 2 и 5 будут выданы.

ii) **participatingEntriesOnly**: Выдаются элементы, отмеченные как участвующие члены, т. е. члены 1, 2, 5, 6, 7, 8 и 9.

iii) **compoundEntry**: Все члены составной статьи будут выданы.

Для случая d) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Члены, отмеченные как содействующие члены, т. е. члены 2 и 5 будут выданы.

ii) **participatingEntriesOnly** и **compoundEntry**: Все члены составной статьи будут выданы.

C.2.3 Пример фильтра 3

Теперь изменим фильтр на {rfc822=cs@cs.dk & service=e-mail}. С параметром **familyGrouping**, установленным в следующее состояние:

a) **entryOnly**: Только один член 5 соответствовал бы фильтру и был бы отмечен как содействующий член и как участвующий член.

b) **strands**: Любая трасса, проходящая через член 5, соответствовала бы фильтру. Член 5 был бы отмечен как содействующий член. Члены 1, 5, 6, 7, 8 и 9 были бы отмечены как участвующие члены.

c) **multiStrand**: Любая трасса, проходящая через член 5, вместе с любой трассой семейства почтовых адресов, соответствовала бы фильтру. Член 5 был бы отмечен как содействующий член. Члены 1, 2, 3, 5, 6, 7, 8 и 9 были бы отмечены как участвующие члены.

d) **compoundEntry**: Член 5 соответствовал бы фильтру и был бы отмечен как содействующий член. Все члены отмечены как участвующие члены.

Для случая a) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly** и **participatingEntriesOnly**: Элемент 5 будет выдан.

ii) **compoundEntry**: Все члены составной статьи будут выданы.

Для случая b) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Член 5 будет выдан.

ii) **participatingEntriesOnly**: Выдаются все члены, отмеченные как участвующие члены, т. е. члены 1, 5, 6, 7, 8 и 9.

iii) **compoundEntry**: Все члены составной статьи будут выданы.

Для случая c) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Член 5 будет выдан.

ii) **participatingEntriesOnly**: Выдаются все члены, отмеченные как участвующие члены, т. е. члены 1, 2, 3, 5, 6, 7, 8 и 9.

iii) **compoundEntry**: Все члены составной статьи будут выданы.

Для случая d) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Член 5 будет выдан.

ii) **participatingEntriesOnly** и **compoundEntry**: Все члены составной статьи будут выданы.

C.2.4 Пример фильтра 4

Наконец, изменим фильтр на {cn=Smith & givenName=Charles}. Только один порождающий член соответствовал бы фильтру.

a) **entryOnly**: Только порождающий член (член 1) был бы отмечен как содействующий член и как участвующий член.

b) **strands, multiStrand** и **compoundEntry**: Порождающий член был бы отмечен как содействующий член, и все члены были бы отмечены как участвующие члены.

Для случая a) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly** и **participatingEntriesOnly**: Член 1 будет выдан.

ii) **compoundEntry**: Все члены составной статьи будут выданы.

Для случая b) выдаваемая информация зависела бы от характеристик семейства в части выдачи:

i) **contributingEntriesOnly**: Член 1 будет выдан.

ii) **participatingEntriesOnly** и **compoundEntry**: Все члены составной статьи будут выданы.

Приложение D

Поправки и исправления

(Данное Приложение не является составной частью настоящей Рекомендации | Международного стандарта)

Данное издание настоящей спецификации Справочника включает в себя следующие проекты поправок к предыдущему изданию, по которым в ИСО/МЭК состоялось голосование и утверждение:

- Поправку 1 о расширениях для поддержки результатов в виде страниц на DSP.
- Поправку 2 о расширениях для поддержки концепции атрибутов "друзей".
- Поправку 3 о максимальном приведении в соответствие X.500 и LDAP.

Данное издание этой спецификации Справочника включает в себя технические исправления, которые учитывают следующие Сообщения о дефектах: 308, 309, 313 и 316.

СЕРИИ РЕКОМЕНДАЦИЙ МСЭ-Т

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

Отпечатано в Швейцарии
Женева, 2006 г.