МЕЖДУНАРОДНЫЙ СОЮЗ ЭЛЕКТРОСВЯЗИ

МСЭ-Т X.518

СЕКТОР СТАНДАРТИЗАЦИИ
ЭЛЕКТРОСВЯЗИ МСЭ

(08/2005)

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

Справочник

Информационные технологии – Взаимосвязь открытых систем – Справочник: Процедуры распределенных операций

Рекомендация МСЭ-Т X.518

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 9594-4
РЕКОМЕНДАЦИЯ МСЭ-Т X.518

Информационные технологии – Взаимосвязь открытых систем – Справочник: Процедуры распределенных операций

Резюме

В настоящей Рекомендации | Международном стандарте определяются процедуры, с помощью которых распределенные компоненты Справочника взаимодействуют для того, чтобы предоставить пользователям соответствующую услугу.

Источник

Рекомендация МСЭ-Т X.518 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 9594-4.

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 по адресу: http://www.itu.int/ITU-T/ipr/.

СОДЕРЖАНИЕ

Стр.

Стр.

Стр.

Введение

Настоящая Рекомендация | часть Международного стандарта вместе с другими Рекомендациями | Международными стандартами разработана для упрощения взаимосвязи систем обработки информации в целях предоставления справочных услуг. Совокупность таких систем вместе с хранимой ими Справочной информацией можно рассматривать как единое целое, называемое *Справочником*. Хранимая в Справочнике информация, называемая в совокупности информационной базой Справочника (DIB), обычно используется для содействия обеспечению связи между объектами, с объектами или относительно объектов, примерами которых могут служить объекты прикладного уровня, люди, терминалы и списки рассылки.

Справочник играет важную роль во взаимосвязи открытых систем, задачей которой является обеспечить взаимосвязи систем обработки информации, используя для этого лишь минимальные технические соглашения, выходящие за рамки самих стандартов взаимосвязи.

- от различных производителей;
- находящихся под различным управлением;
- относящихся к различным уровням сложности; и
- различного возраста.

Данная Рекомендация | Международный стандарт определяет процедуры, которые позволяют распределенным компонентам Справочника взаимодействовать с целью оказания пользователям соответствующих услуг.

Данная Рекомендация | Международный стандарт предоставляет фундаментальные структуры, на основе которых другие группы стандартов и отраслевые форумы могут определять собственные индустриальные профили. Многие из функций, определенных как необязательные в данных фундаментальных структурах, могут при определенных условиях становиться обязательными при их реализации в рамках профилей. В данном пятом издании этой Рекомендации | Международного стандарта редактируется и расширяется, однако не заменяется четвертое издание этой Рекомендации | Международного стандарта. Реализации могут по-прежнему объявлять о соответствии четвертому изданию. Однако, начиная с некоторого момента времени четвертое издание уже не будет поддерживаться (т. е. обнаруженные дефекты уже не могут быть исправлены). Рекомендуется, чтобы реализации начали как можно быстрее соответствовать пятому изданию.

В данном пятом издании описываются версии 1 и 2 из протоколов Справочника.

В первом и втором изданиях определялась только версия 1. Большая часть услуг и протоколов, определенных в настоящем издании, спроектированы для работы согласно версии 1. Однако некоторые усовершенствованные услуги и протоколы, например подписанные параметры ошибки, не будут функционировать, если не все объекты Справочника, участвующие в конкретной операции, согласованы с версией 2. Независимо от версии, с которой обеспечено согласование, различия между услугами и между протоколами, определенными в пятом издании, за исключением специально предназначенных для версии 2, улаживаются при использовании правил расширяемости, определенных в Рекомендации МСЭ-Т X.519 | ИСО/МЭК 9594-5.

Приложение A, которое является неотъемлемой частью данной Рекомендации | Международного стандарта, содержит модуль ASN.1 для распределенных операций Справочника.

Приложение B, которое не является неотъемлемой частью данной Рекомендации | Международного стандарта, описывает пример распределенного разрешения имени.

Приложение C, которое не является неотъемлемой частью данной Рекомендации | Международного стандарта, описывает аутентификацию в условиях распределенных операцияй.

Приложение D, которое является неотъемлемой частью данной Рекомендации | Международного стандарта, предоставлять определения классов информационного объекта ASN.1, которые вводятся в данной спецификации Справочника.

Приложение E, которое не является неотъемлемой частью данной Рекомендации | Международного стандарта, иллюстрирует поддержание знаний.

Приложение F, которое не является неотъемлемой частью данной Рекомендации | Международного стандарта, содержит перечень поправок и сообщений о дефектах, которые были включены для составления данного издания Рекомендации | Международного стандарта.

МЕЖДУНАРОДНЫЙ СТАНДАРТ
РЕКОМЕНДАЦИЯ МСЭ-Т

Информационные технологии – Взаимосвязь открытых систем – Справочник: процедуры распределенных операций

РАЗДЕЛ 1 – ОБЩИЕ ПОЛОЖЕНИЯ

1 Сфера применения

Данная Рекомендация | Международный стандарт определяет поведение системных агентов Справочника (DSA), которые принимают участие в распределенном приложении Справочника. Разрешенное поведение было спроектировано таким образом, чтобы обеспечить соответствующую услугу при условии широкого распределения информационной базы Справочника (DIB) по многим DSA.

Предполагается, что Справочник не является системой базы данных общего назначения, хотя он может быть создан на основе таких систем. Предполагается также, что частота запросов значительно превышает частоту вносимых обновлений.

2 Нормативные справочные документы

Указанные ниже Рекомендации и Международные стандарты содержат положения, которые путем ссылки на них в данном тексте составляют положения настоящей Рекомендации. На момент публикации указанные издания были действующими. Все Рекомендации и стандарты могут подвергаться пересмотру; поэтому сторонам соглашений, основанных на данной Рекомендации | Международном стандарте, предлагается изучить возможность применения последнего издания Рекомендаций и стандартов, перечисленных ниже. 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

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The basic model.*
- Рекомендация МСЭ-Т X.500 (2005 г.) | ИСО/МЭК 9594-1:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Обзор понятий, моделей и услуг*.
- Рекомендация МСЭ-Т X.501 (2005 г.) | ИСО/МЭК 9594-2:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Модели*.
- ITU-T Recommendation X.509 (2005) | ISO/IEC 9594-8:2005, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks.*
- Рекомендация МСЭ-Т X.511 (2005 г.) | ИСО/МЭК 9594-3:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Определение абстрактной службы.*
- ITU-T Recommendation X.519 (2005) | ISO/IEC 9594-5:2005, *Information technology – Open Systems Interconnection – The Directory: Protocol specifications.*
- 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 объектные классы.*
- Рекомендация МСЭ-Т X.525 (2005 г.) | ИСО/МЭК 9594-9:2005, *Информационные технологии – Взаимосвязь открытых систем – Справочник: Копирование.*
- ITU-T Recommendation X.530 (2005) | ISO/IEC 9594-10:2005, *Information technology – Open Systems Interconnection – The Directory: Use of systems management for administration of the Directory.*
- ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

- ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*

2.2 Другие справочные документы

- IETF RFC 2251 (1997), *Lightweight Directory Access Protocol (v3).*
- IETF RFC 3377 (2002), *Lightweight Directory Access Protocol (v3): Technical Specification.*

3 Определения

Для целей настоящей Рекомендации | Международного стандарта используются следующие определения.

3.1 Определение модели передачи данных

В Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5 определяется следующий термин:

a) *заголовок прикладной записи.*

3.2 Основные определения Справочника

В Рек. МСЭ-Т X.500 | ИСО/МЭК 9594-1 определяются следующие термины:

a) *Справочник*;

b) *информационная база Справочника.*

3.3 Определения модели Справочника

В Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2 определяются следующие термины:

a) *точка доступа*;

b) *псевдоним*;

c) *выделенное имя*;

d) *информационное дерево Справочника*;

e) *системный агент Справочника*;

f) *пользовательский агент Справочника*;

g) *относительное отличительное имя*.

3.4 Определения информационной модели DSA

В Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2 определяются следующие термины:

a) *категория*;

b) *повсеместно используемый*;

c) *контекстный префикс*;

d) *перекрестная ссылка*;

e) *фрагмент DIB*;

f) *информационное дерево DSA*;

g) *зависящая от DSA статья*;

h) *тип DSE*;

i) *непосредственно вышестоящая ссылка*;

j) *информация знаний*;

k) *категория ссылки на знание*;

l) *тип ссылки на знание*;

m) *контекст именования*;

n) *неспецифическое знание*;

o) *неспецифическая подчиненная ссылка*;

p) *операционный атрибут*;

q) *траектория ссылки*;

r) *специфическое знание*;

s) *подчиненная ссылка*;

t) *вышестоящая ссылка*.

3.5 Определения абстрактной службы

В Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3 определяется следующий термин:

a) *потоковый результат.*

3.6 Определения репликации Справочника

В Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9:

a) *полнота атрибута*;

b) *теневое операционное связывание*;

c) *полнота подчиненного*;

d) *единица репликации*.

3.7 Определение распределенных операций

В данной Рекомендации | Международном стандарте определяются следующие термины:

3.7.1 базовый объект: Объект или статья псевдонима, который является целью операции, указанным инициатором при ее создании этой операции.

3.7.2 граничащий DSA: Тот DSA, с которым выполнил операцию Bind (привязывание) Пользовательский агент Справочника DUA, который организует запрос.

3.7.3 результаты, разбитые на страницы граничащим DSA: Разбиение на страницы полностью выполняется тем DSA, с которым связан DUA.

ПРИМЕЧАНИЕ. – Это является единственным режимом разбиения на страницы, который поддерживается системами, которые соответствуют изданиям, вышедшим ранее пятого издания.

3.7.4 связывание: Общий термин для одиночного и множественного связывания

3.7.5 информация контекстного префикса: Операционная и пользовательская информация, которая передается вышестоящим DSA подчиненному DSA с помощью RHOB и относится к вершинам дерева DIT, которые являются вышестоящими для подчиненного контекстного префикса.

3.7.6 распределенное разрешение имени: Процесс, с помощью которого разрешение имени выполняется в более чем одном DSA.

3.7.7 DSP постраничные результаты: Условия протокола DSP, когда исполняющим DSA является DSA, отличный от граничащего DSA, и когда постраничные результаты предоставляются первоначальным исполнителем.

3.7.8 ошибка: Информация, которая пересылается от исполнителя к заказчику и которая сообщает об отрицательном результате выполнения полученного перед этим запроса.

3.7.9 тяжелая ошибка: Определенная ошибка, которая сообщает о том, что операция в данный момент не может выполняться без внешнего вмешательства.

3.7.10 иерархическое операционное связывание (HOB): Отношение между двумя главными DSA, которые содержат контексты именования, один из которых является непосредственным подчиненным для другого, в котором вышестоящий DSA содержит подчиненную ссылку на подчиненный DSA.

3.7.11 первоначальный исполнитель: Первый DSA, который начинает выполнение, т. е. первый DSA, который вступает в фазу оценки данной операции.

3.7.12 операции изменения: Это операции изменения Справочника, к которым относятся Modify Entry (изменить статью), Add Entry (добавить статью), Remove Entry (удалить статью) и Modify DN.

3.7.13 множественное связывание: Режим взаимодействия, в котором обрабатывающий запрос DSA самостоятельно отправляет множество запросов параллельным или последовательным способом к набору других DSA.

3.7.14 операция, которая обращается к нескольким статьям: Это операции поиска Справочника, т. е. операции List и Search.

3.7.15 разрешение имени: Процесс определения местонахождения статьи с помощью последовательного сопоставления каждого RDN в потенциальном имени с вершинами дерева DIT.

3.7.16 неспецифическое операционное связывание (NHOB): Взаимосвязь между двумя главными DSA, которые хранят контексты именования, один из которых является непосредственным подчиненным для другого, в которой вышестоящий DSA хранит неспецифическую подчиненную ссылку на подчиненный DSA.

3.7.17 декомпозиция NSSR: Декомпозиция неспецифических ссылок на знания на подзапросы, которые будут выполняться другими DSA. Эти подзапросы могут либо связываться с другими DSA тем DSA, который выполняет декомпозицию, либо для обработки заказчику возвращается ссылка продолжения, которая идентифицирует другие DSA, либо осуществляющий декомпозицию DSA может выполнить некоторые из подзапросов, оставляя другие подзапросы неизученными для выполнения заказчиком.

3.7.18 прогресс операции: Набор значений, которые определяют степень выполнения разрешения имени.

3.7.19 инициатор: DUA, который инициировал какую-то (распределенную) операцию.

3.7.20 разбиение на страницы: Результат операций **search** или **list**, который возвращается отдельными фрагментами в форме одной или нескольких страниц, каждая такая страница содержит ограниченное количество статей.

3.7.21 исполнитель: DSA, который получает запрос (на выполнение операции).

ПРИМЕЧАНИЕ. – Исполнителем также является начальный исполнитель за тем возможным исключением, когда операции используют несколько DSA для оценки.

3.7.22 процедура: Спецификация (неформальная) того, как DSA отображает заданный набор входных аргументов и собственное дерево информации DSA на результаты.

ПРИМЕЧАНИЕ. – Входные аргументы и результаты могут соответствовать информации, полученной в запрошенной операции и информации, переданной в ответ, или же может представлять промежуточные этапы в вычислении ответа из запрошенной операции. В п. 14.2 первый набор входных аргументов и результатов называется внешним.

3.7.23 релевантное иерархическое операционное связывание (RHOB): В зависимости от контекста HOB или NHOB.

3.7.24 реферал: Результат, который может возвращаться DSA, который не может выполнить операцию самостоятельно, и который идентифицирует один или более других DSA, более подготовленных к выполнению этой операции.

3.7.25 ответ: Результат или ошибка.

3.7.26 запрос: Информация, которая состоит из кода операции и связанных аргументов, и служит для передачи операции Справочника от заказчика к исполнителю.

3.7.27 декомпозиция запроса: Декомпозиция запроса на подзапросы, которые будут выполняться другими DSA. Эти подзапросы могут либо связываться с другими DSA тем DSA, который выполняет декомпозицию, либо для обработки заказчику возвращается ссылка продолжения, которая идентифицирует другие DSA, либо осуществляющий декомпозицию DSA может выполнить некоторые из подзапросов, оставляя другие подзапросы неизученными для выполнения заказчиком.

3.7.28 заказчик: DUA или DSA, которые отправляют запрос на выполнение (вызов) операции.

3.7.29 операция, **которая обращается к одной статье**: Это операции чтения Справочника, т. е. операции Read и Compare.

3.7.30 мягкая ошибка: Ошибка, которая может иметь временный характер, или которая может сообщать о локализованной проблеме, в этом случае использование отличающейся ссылки на знание или другой точки доступа может обеспечить результат или привести к получению тяжелой ошибки.

3.7.31 подчиненный DSA: Для двух DSA, которые использую общее HOB или NHOB, им является тот DSA, который хранит подчиненный контекст именования.

3.7.32 подзапрос: Запрос, который создается при декомпозиции запроса.

3.7.33 вышестоящий DSA: Для двух DSA, между которыми образовано HOB или NHOB, им является тот DSA, который содержит вышестоящий контекст именования.

3.7.34 вышестоящий, подчиненный DSA: Два главных DSA, которые содержат контексты именования, из которых один является непосредственно подчиненным для другого. Взаимосвязь между двумя DSA управляется явным образом с помощью HOB (или NHOB) или же существует в неявном виде благодаря вышестоящему DSA, который содержит подчиненную (или неспецифическую подчиненную) ссылку на подчиненный DSA.

3.7.35 имя целевого объекта: Имя статьи, к которой операция должна быть направлена на определенном этапе разрешения имени или же которое принимает участие в оценке операции.

3.7.36 единичное связывание: Режим взаимодействия, который может в качестве необязательного использоваться DSA, когда он не может выполнить операцию самостоятельно. DSA выполняет *связывание* с помощью вызова операции другого DSA и затем ретранслирует результат первоначальному заказчику.

4 Сокращения

Для целей настоящей Рекомендации | Международного стандарта используются следующие сокращения.

ASN.1	Абстрактно-синтаксическая нотация 1
DISP	Протокол теневого копирования информации Справочника
DMD	Домен управления Справочником
DOP	Протокол управления операционным связыванием Справочника
DSE	Зависящая от DSA статья
HOB	Иерархическое операционное связывание
NHOB	Неспецифическое иерархическое операционное связывание
NSSR	Неспецифическая подчиненная ссылка
RHOB	Релевантное иерархическое операционное связывание

5 Соглашения

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.518 | ИСО/МЭК 9594-4. Термин "спецификация 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 Рекомендаций МСЭ-Т X.509 и частям 1–10 издания стандарта ИСО/МЭК 9594:2001.

В настоящей спецификации Справочника используется термин *системы пятого издания* для ссылки на системы, соответствующие пятому изданию спецификаций Справочника, т. е. изданиям 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. Названия процедур, упоминаемых при определении семантики обработки, выделяются в тексте полужирным шрифтом Times. Разрешения на управление доступом предоставляются курсивом шрифта Times.

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

РАЗДЕЛ 2 – ОБЗОР

6 Обзор

Абстрактная служба Справочника позволяет запрашивать, извлекать и изменять информацию Справочника в дереве DIB. Эта служба описывается с помощью абстрактного объекта Справочника, как это указывается в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Аналогично, протокол Lightweight Directory Access Protocol (LDAP) обеспечивает запрос, извлечение и изменение информации Справочника в дереве DIB. Данный протокол и службы, которые он обеспечивает, описываются в RFC 3377.

Спецификация абстрактного объекта Справочника ни в коей мере не направлена на решение вопросов, связанных с физической реализацией Справочника: в частности, она не затрагивает спецификацию системных агентов Справочника (DSA, Directory System Agents), в которых хранится и которыми управляется дерево DIB и с помощью которых обеспечивается служба. Более того, она не затрагивает то, является ли DIB централизованным (т. е. содержится в одном DSA), или же распределяется по многим DSA. Следовательно, в описании службы также не содержится требований к DSA относительно знания, способности перемещаться или же взаимодействовать с другими DSA с целью поддержки абстрактной службы в распределенном окружении.

Данная спецификация Справочника уточняет свойства абстрактного объекта Справочника, это производится с помощью набора из одного или более DSA, которые совместно образуют распределенную службу Справочника.

В дополнение к этому, данная спецификация Справочника определяет разрешенные способы, которые позволяют распределить DIB по одному или нескольким DSA. Для предельного случая, когда DIB содержится в единственном DSA, Справочник является централизованным; если же DIB распределен по двум или более DSA, то специфицируются знание и механизмы навигации, которые гарантируют, что DIB потенциально доступен для всех DSA, которые содержат образующие статьи.

Отдельные части DIB также могут дублироваться в нескольких DSA. Протоколы, которые описываются в данной спецификации Справочника, позволяют использовать дублированную информацию для улучшения доступности, производительности и эффективности распределенной службы Справочника. Использование дублированной информации в какой-то степени происходит под контролем пользователя, который использует для этого опции управления службой. Описываемые в данной спецификации Справочника процедуры также сообщают о тех возможностях, которые могут использоваться для оптимизации проекта с помощью дублирования информации.

Кроме этого, определяются запросы для обеспечения взаимодействия, которые позволяют пользователям контролировать определенные операционные характеристики Справочника. В частности, пользователь может управлять тем, что DSA, который отвечает на запрос Справочника относительно информации, которая хранится в другом(их) DSA), может либо непосредственно обратиться к другим DSA (связывание) или же ответить информацией о других DSA, которые могут далее обрабатывать запрос (реферал).

В общем случае решение о связывании или реферале DSA принимает, руководствуясь установленным пользователем управлением службой, а также собственными административными, операционными или техническими условиями, в которых находится DSA.

Понимая, что в общем случае Справочник является распределенным, и что запросы к Справочнику будут выполняться произвольным количеством взаимодействующих между собой DSA, которые в соответствии с указанными выше критериями могут выполнять связывание или использовать реферал, данная спецификация Справочника определяет соответствующие процедуры, которые DSA должен выполнять в ответ на распределенный запрос к Справочнику. Эти процедуры гарантируют, что пользователи службы распределенного Справочника будут воспринимать его как дружественный по отношению к пользователю и согласованный.

РАЗДЕЛ 3 – МОДЕЛИ РАСПРЕДЕЛЕННОГО СПРАВОЧНИКА

7 Модель системы распределенного Справочника

Определенная в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3 абстрактная служба Справочника моделирует Справочник как объект, который обеспечивает для своих пользователей набор услуг Справочника. Пользователи Справочника получают доступ к службам с помощью точки доступа. Справочник может иметь одну или более точек доступа, каждая из точек доступа определяется услугами, которые она предоставляет, а также способом взаимодействия, с помощью которого обеспечиваются эти службы.

На рис. 1 показана модель распределенного Справочника, которая будет использоваться в качестве основы для спецификации распределенных аспектов Справочника. Она представляет Справочник как состоящий из набора одного или нескольких DSA.

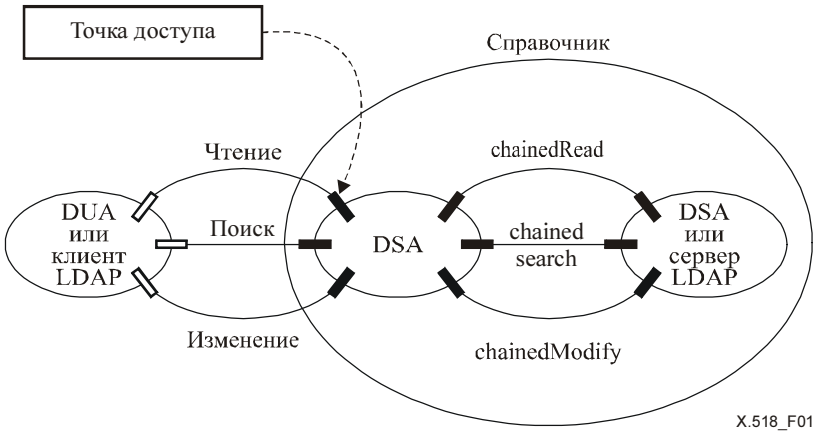


Рисунок 1 – Объекты модели распределенного Справочника

В деталях DSA специфицируются в следующих разделах спецификации Справочника. В данном разделе мы просто определяем некоторые из его характеристик, для того чтобы он мог послужить также в качестве введения, а также мы устанавливаем отношения между данной спецификацией Справочника и другими спецификациями Справочника.

Целью определения DSA является обеспечить возможности для адаптации распределения DIB, а также обеспечить, чтобы некоторое количество физически распределенных DSA могли бы взаимодействовать в заранее предписанной, кооперативной манере для того, чтобы обеспечить службы Справочника для его пользователей (DUA или клиентов LDAP).

Рис. 1 иллюстрирует взаимоотношения между абстрактной службой Справочника и абстрактной службой DSA. Абстрактная служба Справочника определена в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3 и обеспечивается с помощью некоторого количества операций Справочника. Для того чтобы реализовать такую службу, образующие Справочник DSA должны взаимодействовать между собой. Природа такого взаимодействия определяется с помощью тех услуг, которые DSA способен предоставить другому DSA – это является абстрактной службой DSA. Абстрактная служба DSA обеспечивается с помощью некоторого набора операций, которые носят название chained operations (связанные операции), каждая из таких операций имеет аналог в абстрактной службе Справочника. Таким образом, заданная операция в абстрактной службе Справочника, например, Read (чтение), может потребовать, чтобы DSA, который обеспечивает данную услугу, взаимодействовал с одним или более DSA с использованием связанных операций – например, Chained Read (связанное чтение).

ПРИМЕЧАНИЕ. – Также возможно для DSA, что существуют запросы LDAP о связывании операций, например, с использованием управления LDAP или расширенных операций; однако необходимые для этого процедуры и протоколы выходят за рамки данной спецификации Справочника.

8 Модель взаимодействия DSA

Основной характеристикой Справочника является то, что при наличии распределенной DIB, любой запрос к службе со стороны пользователя должен быть удовлетворен (при условии соблюдения требований к безопасности, контролю доступа и административной стратегии) независимо от точки доступа, с помощью который был подан этот запрос. При реализации данного требования необходимо, чтобы любой DSA, который принимает участие в обеспечении определенной службы, имел бы определенное знание (согласно Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2) о том, где находится запрашиваемая информация, и тогда либо возвращает эту информацию тому, кто ее запрашивает, или же пытается выполнить запрос от своего имени. (В качестве заказчика службы может выступать DUA, клиент LDAP или же другой DSA: в последнем случае оба DSA должны поддерживать DSP.)

Для того чтобы удовлетворить этим требованиям, определяются три режима взаимодействия между DSA, а именно: "uni-chaining" (одиночное связывание), "multi-chaining" (множественное связывание) и "referral" (реферал). На протяжении всего дальнейшего рассмотрения данной спецификации Справочника общее понятие

"Связывание" используется, в зависимости от контекста, для обозначения одиночного связывания или множественного связывания. "Связывание" относится к попытке DSA выполнить запрос путем посылки одной или многих связанных операций другим DSA; в то время как "реферал" (направление) относится к возвращению заказчику информации о знании, после чего заказчик может самостоятельно взаимодействовать с теми DSA, которые представлены в информации о знании.

Одиночное связывание или взаимодействие с помощью реферала может возникать в результате единственного запроса. Верно и обратное, запрос можно представить в виде нескольких подзапросов и только затем можно переходить к рассмотрению взаимодействия. В результате декомпозиции запроса могут возникать взаимодействия с помощью множественного связывания или рефералов, или же с помощью двух этих методов. Определяются два типа декомпозиции; декомпозиция NSSR и декомпозиция запроса.

8.1 Декомпозиция запроса

8.1.1 Декомпозиция NSSR

Декомпозиция NSSR представляет собой процесс подготовки идентичных запросов, которые готовы к отправке (либо последовательно, либо параллельно) к нескольким подчиненным DSA – это происходит, когда при разрешении имени встречается NSSR. Неспецифические подчиненные ссылки не содержат RDN для указываемых подчиненных контекстов именования, таким образом, ссылающийся DSA не может сказать, какой из подчиненных DSA содержит определенный(ые) подчиненный(ые) контекст(ы) именования. В процессе разрешения имени DSA, который встречает NSSR, должен отправить идентичный запрос каждому подчиненному DSA (при отсутствии теневого копирования). Это может производиться последовательно или параллельно. Обычно только один DSA способен продолжать разрешение имени; остальные возвращают ошибку **serviceError** (ошибка службы) с проблемой **unableToProceed** (невозможно продолжать обработку). В определенных (достаточно редко встречающихся) обстоятельствах существует возможность, что разрешение имени продолжат несколько DSA, что приводит к дублированию результатов.

ПРИМЕЧАНИЕ. – NSSR не могут ссылаться на серверы LDAP.

8.1.2 Декомпозиция запроса

Другая форма представления запроса – декомпозиция запроса – является процессом, который выполняется DSA внутренним образом, до начала коммуникаций с одним или несколькими другими серверами DSA и/или LDAP. Запрос представляется в виде нескольких подзапросов, возможно, различающихся между собой, при этом каждый из таких подзапросов выполняет часть первоначальной задачи. Декомпозиция запроса может использоваться только при оценке операций List (список) или Search (поиск). После декомпозиции запроса каждый из подзапросов с целью выполнения задачи может быть связан с другими серверами DSA и/или LDAP, или же заказчику могут быть возвращены частичные результаты (в виде встроенного реферала). Примером того, как один подзапрос может генерироваться для различных серверов DSA и/или LDAP, является случай, когда статья имеет подчиненные ссылки и/или NSSR, которые совместно ссылаются на более чем один сервер DSA или LDAP. Примером того, как различные подзапросы могут генерироваться для одинаковых или различных серверов DSA и/или LDAP, является случай, когда во время поиска (Search) одного поддерева встречаются две различные записи, и каждая из них имеет подчиненную ссылку.

8.2 Одиночное связывание

Данный режим взаимодействия (представленный на рис. 2) может использоваться одним DSA для отправки запроса другому DSA в том случае, когда первый имеет информацию о контекстах именования, которые содержатся во втором. Одиночное связывание может использоваться для контакта с другим, единственным DSA, на который существует перекрестная ссылка, подчиненная ссылка, вышестоящая ссылка, ссылка поставщика или главная ссылка.

ПРИМЕЧАНИЕ. – На рис. 2 порядок взаимодействия определяется числами, которые связаны с линиями взаимодействия.

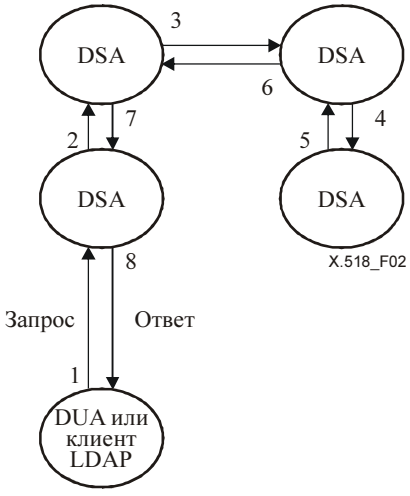


Рисунок 2 – Режим одиночного связывания

8.3 Множественное связывание

Данный режим взаимодействия используется DSA для передачи нескольких исходящих запросов, причиной которых послужил единственный входящий запрос, что случается в результате декомпозиции запроса или декомпозиции NSSR.

8.3.1 Параллельное множественное связывание

При помощи параллельного множественного связывания DSA одновременно передает несколько исходящих запросов (см. рис. 3a). Хотя использование параллельного множественного связывания может привести к улучшению производительности, при определенных условиях, например, в присутствии теневого копирования, оно может приводить к получению дублирующих друг друга результатов.

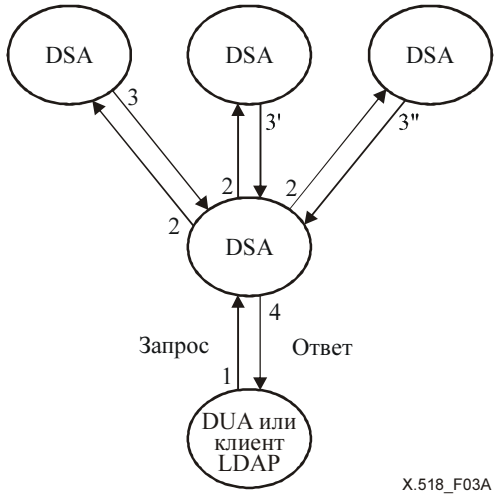
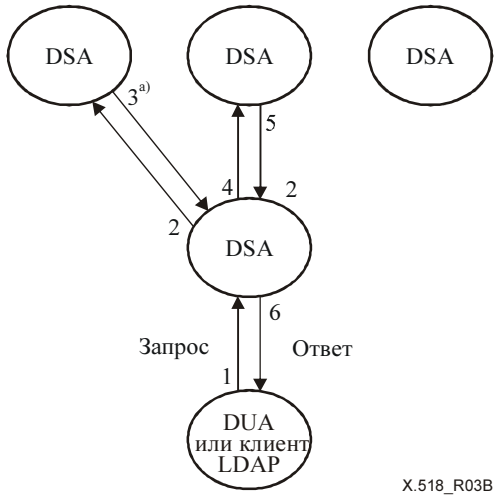


Рисунок 3a – Параллельное множественное связывание

8.3.2 Последовательное множественное связывание

При последовательном множественном связывании DSA передает за один раз один исходящий запрос, а затем ожидает результата или сообщения об ошибке – только после этого отправляется следующий запрос (см. рис. 3b). Хотя последовательное множественное связывание может и не являться наиболее быстрым режимом взаимодействия, оно уменьшает вероятность получения дублирующих результатов.

ПРИМЕЧАНИЕ. – DSA может использовать комбинацию параллельного множественного связывания и последовательного множественного связывания.



[a)] Обработка невозможна.

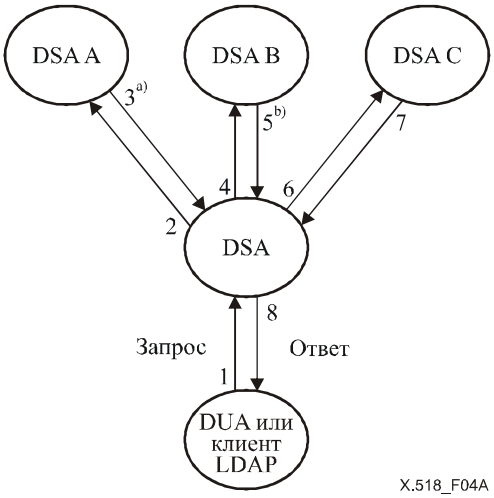
Рисунок 3b – Последовательное множественное связывание (в результате декомпозиции NSSR)

8.4 Реферал

Реферал (см. рис. 4a и 4b) DSA возвращает в ответ на запрос со стороны DUA, клиента LDAP или другого DSA. Реферал может представлять собой весь ответ (в этом случае он характеризуется как ошибка) или же представлять только часть ответа. Реферал содержит ссылку на знание, которое может являться вышестоящей, подчиненной, перекрестной, неспецифической подчиненной ссылкой, ссылкой поставщика или главной ссылкой.

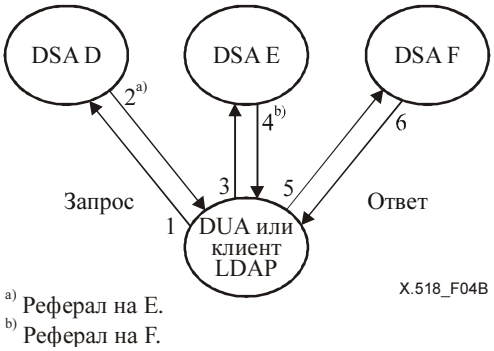
Тот DSA (рис. 4a), который получает реферал, может использовать содержащуюся в нем ссылку на знание, для того чтобы последовательно связать или же осуществить многоадресную передачу (в зависимости от типа ссылки) первоначального запроса другим DSA. Альтернативным образом, когда DSA получает реферал, он может в свою очередь возвратить этот реферал назад в качестве ответа. DUA или клиент LDAP (рис. 4b), который получает реферал, может использовать его для контакта с одним или несколькими другими DSA или же может продвигать запрос далее.

ПРИМЕЧАНИЕ. – На рис. 4a и 4b порядок взаимодействия определяется числами, которые связаны с линиями взаимодействия.



a) Реферал на B.
b) Реферал на C.

Рисунок 4a – Режим реферала (DSA действует на рефералы)



a) Реферал на E.
b) Реферал на F.

Рисунок 4b – Режим реферала (DUA действует на рефералы)

8.5 Определение режима

Если DSA не может самостоятельно полностью разрешить данный запрос, то он должен связать этот запрос (или же запрос, который получен в ходе декомпозиции первоначального запроса) с другим DSA, если только не выполняются следующие условия:

a) связывание запрещено пользователем с помощью управления службами, в этом случае DSA возвращает реферал или **serviceError** с проблемой **chainingRequired**; или же

b) DSA опирается на административные, операционные или другие причины при отказе от связывания, в этом случае DSA возвращает реферал.

ПРИМЕЧАНИЕ 1. – "Технической причиной", которая заставляет отказаться от связывания, является то, что определенный в ссылке знания DSA не поддерживает DSP.

ПРИМЕЧАНИЕ 2. – Если установлено управление службой **localScope**, то DSA (или DMD) должен либо выполнить разрешение данного запроса, либо возвратить ошибку.

ПРИМЕЧАНИЕ 3. – Если пользователь предпочитает использовать рефералы, то он должен установить **chainingProhibited**.

РАЗДЕЛ 4 – АБСТРАКТНАЯ СЛУЖБА DSA

9 Обзор абстрактной службы DSA

Служба Справочника подробно описывается в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Когда, как это моделируется в п. 7, такая услуга предоставляется в распределенных условиях, то можно считать, что она обеспечивается с помощью набора DSA (см. рис. 1).

Для каждой операции, которая определяется в службе Справочника, в абстрактной службе DSA также определяется соответствующая "связанная" операция для использования между DSA, взаимодействующими при выполнении операций данной службы Справочника. Таким образом DSA, который получает операцию Read (чтение) от DUA, может потребовать помощи от другого DSA (например, DSA который содержит нужную запись или ее копию) для выполнения операции, и тогда он посылает данному DSA связанную операцию Chained Read (связанное чтение).

Типы информации, которыми обмениваются в абстрактных службах DSA, определяются в п. 10. Операции и ошибки, относящиеся к абстрактной службе DSA, определяются в пп. 11–13.

10 Типы информации

10.1 Введение

В данном разделе описываются, а в некоторых случаях и определяются, определенное количество типов информации, которые затем используются при определении различных операций для абстрактной службы DSA. Затрагиваемые типы информации имеют отношение к более чем одной операции, которые должны произойти в будущем, или которые являются достаточно сложными или автономными, что позволяет рассматривать отдельно от операции, которая их использует.

Некоторые из типов информации, которые используются при определении абстрактной службы DSA, определяются в других документах. Эти типы приводятся в п. 10.2 и также указывается источник, в котором они определены. Подпункты 10.3–10.10 служат для идентификации и определения типа информации.

10.2 Типы информации, которые определяются в других местах

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

- **aliasedEntryName**;
- **DistinguishedName**;
- **Name**;
- **RelativeDistinguishedName**.

Следующие типы информации определяются в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3:

(Bind (приязывание))

- **DirectoryBind**

(Operations (операции))

- **Abandon**

(Errors (ошибки))

- **abandoned**;
- **attributeError**;
- **nameError**;
- **securityError**;
- **serviceError**;
- **updateError**.

(Information Object Class (класс информационного объекта))

- **OPTIONALLY-PROTECTED**

(Data Type (тип данных))

- **SecurityParameters**

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

– **PresentationAddress**.

10.3 Аргументы связывания

Аргументы связывания **ChainingArguments** представлены в каждой операции связывания, они передают DSA ту информацию, которая необходима для успешного выполнения порученной части общей задачи:

```
ChainingArguments::= SET {
        originator                [0]   DistinguishedName OPTIONAL,
        targetObject              [1]   DistinguishedName OPTIONAL,
        operationProgress         [2]   OperationProgress
                                        DEFAULT { nameResolutionPhase notStarted },
        traceInformation          [3]   TraceInformation,
        aliasDereferenced         [4]   BOOLEAN DEFAULT FALSE,
        aliasedRDNs               [5]   INTEGER OPTIONAL,
                        -- присутствует только в системах, относящихся к первому изданию
        returnCrossRefs           [6]   BOOLEAN DEFAULT FALSE,
        referenceType             [7]   ReferenceType DEFAULT superior,
        info                      [8]   DomainInfo OPTIONAL,
        timeLimit                 [9]   Time OPTIONAL,
        securityParameters        [10]  SecurityParameters DEFAULT { },
        entryOnly                 [11]  BOOLEAN DEFAULT FALSE,
        uniqueIdentifier          [12]  UniqueIdentifier OPTIONAL,
        authenticationLevel       [13]  AuthenticationLevel OPTIONAL,
        exclusions                [14]  Exclusions OPTIONAL,
        excludeShadows            [15]  BOOLEAN DEFAULT FALSE,
        nameResolveOnMaster       [16]  BOOLEAN DEFAULT FALSE,
        operationIdentifier       [17]  INTEGER OPTIONAL,
        searchRuleId              [18]  SearchRuleId OPTIONAL,
        chainedRelaxation         [19]  MRMapping OPTIONAL,
        relatedEntry              [20]  INTEGER OPTIONAL,
        dspPaging                 [21]  BOOLEAN DEFAULT FALSE,
        nonDapPdu                 [22]  ENUMERATED { ldap (0) } OPTIONAL,
        streamedResults           [23]  INTEGER OPTIONAL
        excludeWriteableCopies    [24]  BOOLEAN DEFAULT FALSE }

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

Смысл и значение различных компонентов приводится ниже:

a) Компонент **originator** (инициатор) передает название (первоначального) инициатора данного запроса, если только это не было уже определено в параметрах безопасности. Если в общих аргументах (**CommonArguments**) присутствует **requester** (заказчик), то этот аргумент может быть опущен.

   ПРИМЕЧАНИЕ 1. – В том случае, если инициатор имеет альтернативные выделенные имена, которые различаются в зависимости от контекста, то в качестве значения для **originator** должно использоваться первичное отличительное имя (при условии, что оно известно). В противном случае возможно, что аутентификация и контроль доступа, основанные на использовании значения **originator**, не будут работать должным образом.

b) Компонент **targetObject** передает название объекта, к Справочнику которого производится маршрутизация. Роль такого объекта зависит от текущей операции: это может быть объект, над записью которого должна проводиться операция, или же объект, который служит в качестве базового для запросов или подзапросов, включающих множество объектов (например, **chainedList** или **chainedSearch**). Данный компонент может опускаться только в том случае, если он имеет одинаковое значение с параметром объекта или базового объекта в связанной операции, в этом случае данное значение является его подразумеваемым значением.

   Там, где **targetObject** включает RDN, содержащий тип атрибута и пары значений, для которых существует множественные выделенные значения, определяемые контекстом, то RDN, для которых было выполнено разрешение, являются первичными RDN.

c) Компонент **operationProgress** используется для того, чтобы информировать DSA о ходе выполнения операции, и, следовательно, о той роли, которую он должен выполнять в ее совместном исполнении. Информация, которая передается с помощью данного компонента, описывается в п. 10.5.

d) Компонент **traceInformation** используется для предотвращения образования петель между DSA при организации связывания операций. Перед тем, как связывать операцию с другим DSA, DSA добавляет новый элемент к трассировочной информации (trace information). После запроса о выполнении операции, DSA с помощью трассировочной информации выполняет проверку того, что данная операция не приводит к образованию петли. Информация, которая передается с помощью данного аргумента, описывается в п. 10.6.

e) Компонент **aliasDereferenced** имеет значение **BOOLEAN**, которое сообщает о том, были ли на данный момент в процессе распределенного разрешения имени встречены и разыменованы одна или более статей псевдонима. Установленное по умолчанию значение **FALSE** говорит о том, что разыменование статей псевдонима не производилось.

f) Компонент **aliasedRDNs** сообщает о том, как много RDN в составе **targetObject Name** были сгенерированы с помощью атрибутов **aliasedEntryName** одной (или нескольких) статей псевдонима. В том случае, если встречается и разыменовывается статья псевдонима, то компоненту присваивается целое значение. Этот компонент должен присутствовать в том и только в том случае, если компонент **aliasDereferenced** имеет значение **TRUE**.

ПРИМЕЧАНИЕ 2. – Данный компонент используется для обеспечения совместимости с первыми изданиями Справочника. Если DUA (и DSA) созданы на основе более поздних изданий спецификации Справочника, то они должны всегда опускать данный параметр из **CommonArguments** последующего запроса. Таким образом, Справочник не сигнализирует ошибку если псевдоним разыменовывается для последующих псевдонимов.

g) Компонент **returnCrossRefs** представляет логическое значение, которое сообщает о том, необходимо или нет возвращать обратно первоначальному DSA вместе с результатами или реферalом в качестве перекрестной ссылки ссылку на знание, которое использовалось при выполнении распределенной операции. Значение **FALSE**, которое устанавливается по умолчанию, означает что такое знание возвращать не обязательно.

h) Компонент **referenceType** сообщает DSA, которое запросили выполнить операцию, какой тип знания использовался для того, чтобы осуществить к нему маршрутизацию запроса. Таким образом, DSA может определить ошибки в том знании, которое использует вызывающая сторона. Если обнаруживается такая ошибка, то она индицируется с помощью ошибки **serviceError** с проблемой **invalidReference**. Компонент **ReferenceType** полностью описывается в п. 10.7.

ПРИМЕЧАНИЕ 3. – Если компонент **referenceType** отсутствует, то предполагается, что компонент имеет значение **superior**.

i) Компонент **info** используется для передачи специфической для DMD информации между DSA, которые используются при обработке общего запроса. Этот компонент имеет тип **DomainInfo**, который относится к типу без ограничений:

DomainInfo::= ABSTRACT-SYNTAX.&Type

j) Компонент **timeLimit**, если он присутствует, обозначает время, к которому должна быть завершена операция (см. п. 16.1.4.1). До того, как значение **Time** будет использоваться в любой операции сравнения и если синтаксис **Time** был выбран как тип **UTCTime**, необходимо преобразовать обозначение года из двух цифр в значение из четырех цифр следующим образом:

– Если значение из двух цифр лежит в диапазоне от 00 до 49 включительно, то необходимо добавить к этому значению 2000.

– Если значение из двух цифр лежит в диапазоне от 50 до 99 включительно, то необходимо добавить значение 1900.

ПРИМЕЧАНИЕ 4. – Использование **GeneralizedTime** может помешать совместной работе с теми реализациями, которые не имеют представления о возможности выбора между **UTCTime** и **GeneralizedTime**. Определение того, когда можно использовать **GeneralizedTime**, является обязанностью тех, кто определяет область использования спецификации Справочника, например, профильные группы (profiling groups). Ни в коем случае нельзя использовать **UTCTime** для представления даты позднее 2049 года.

k) Компонент **SecurityParameters** определяется в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если он отсутствует, то это эквивалентно пустому набору параметров безопасности.

l) Компонент **entryOnly** устанавливается равным **TRUE** в том случае, если первоначальной операцией являлась операция Search (поиск), в которой аргумент **subset** был установлен как **oneLevel**, а обнаруженная статья псевдонима является непосредственно подчиненной для **baseObject**. При этом DSA, который успешно выполнил разрешение имени для имени **targetObject**, должен выполнить оценку объекта только для указанной статьи.

m) Компонент **uniqueIdentifier** является необязательным и предоставляется тогда, когда необходимо подтвердить имя инициатора. Тип данных **UniqueIdentifier** описывается в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

n) Компонент **authenticationLevel** является необязательным и предоставляется тогда, когда необходимо обозначить способ, с помощью которого проводилась аутентификация. Тип данных **AuthenticationLevel** описывается в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

o) Компонент **exclusions** имеет значение только для операции Search (поиск); он указывает (при условии его наличия), какие из статей, которые относятся к **targetObject**, должны исключаться из результатов операции Search (поиск) (см. п. 10.9).

p) Компонент **excludeShadows** имеет значение только для операций Search и List; он обозначает, что поиск должен применяться к самим статьям, а не к их копиям. Этот компонент является необязательным и может использоваться DSA в качестве одного из способов для предотвращения получения дублирующих результатов (см. п. 20.1).

q) Компонент **nameResolveOnMaster** важен только при выполнении разрешения имени, и устанавливается только в том случае, если было встречено NSSR. Если его значение устанавливается как **TRUE**, то оно сигнализирует о том, что при последующем разрешении имени, т. е. при сопоставлении остающегося RDN из **nextRDNToBeResolved**, нельзя использовать информацию из копии статьи, включая копии с возможностью записи для реализации с многими мастерами. Последующее разрешение каждого из остающихся RDN не должно выполняться в главном DSA для статей, которые определяются данным RDN (см. п. 20.1).

r) Компонент **operationIdentifier** облегчает корреляцию операций DAP с последующими связанными операциями DSP, а также с результатами. Он присваивается тем DSA, который первым получил запрос DAP или же копируется из аргументов связывания запроса DSP, который требует последующего связывания. DSA, который присваивает **operationIdentifier**, не должен повторно использовать присвоенное целое значение в течение достаточно большого промежутка времени. Корреляция связанных запросов и результатов DAP и DSP облегчается при помощи протоколирования DSA для каждой операции и результата значения **operationIdentifier** вместе с названием того DSA, который присвоил это значение (первый DSA в трассировочной информации **traceInformation** для связанного запроса). Такая корреляция может быть полезна для целей протоколирования, аудита, начисления расходов и расчетов, и так далее.

s) Компонент **searchRuleId** передает уникальный идентификатор правила. Он включается тем DSA, который выполняет первоначальную процедуру **Search procedure (I)** в том случае, если данная процедура запускается внутри специфической для данной службы административной области, и операция поиска прогрессирует к другим DSA, или же при обработке вниз по DIT при следовании псевдонимам или при следовании иерархическим указателям групп.

t) Компонент **chainedRelaxation** обеспечивает, чтобы релаксация проводилась распределенным образом с помощью связанных операций поиска. Если DSA получает связанную операцию поиска и при этом поддерживает стратегии релаксации, то он может использовать предоставленный компонент **chainedRelaxation** вместо любой другой стратегии релаксации, которую он может реализовать – благодаря этому обеспечивается координация релаксации с другими DSA, которые потенциально способны возвращать результаты поиска.

u) Элемент **relatedEntry** должен существовать всегда, когда принимающий DSA обязан разрешить относящиеся статьи. При наличии, принимающий DSA должен отвечать только специфическим относящимся элементам статей, которые специфицируются значением **relatedEntry** в **joinAttributes** для **SearchArgument**. Таким образом, значение **relatedEntry**, равное нулю, должно выбирать первый элемент в последовательности **joinAttributes** для **SearchArgument**. Значение никогда не должно превышать количество элементов минус один в компоненте **joinAttributes** для **SearchArgument**. В случае отсутствия элемента **relatedEntry** в **ChainingArguments**, операция DSP, которая специфицирует относящиеся статьи, должна индицировать, что распределенная операция, для которой производится связывание, является базовым поиском, а не является относящейся частью статьи для поиска.

ПРИМЕЧАНИЕ 5. – Если DSA, с которым выполняется связывание, должен обрабатывать как результаты нормального поиска, так и результаты для относящихся статей, то это выполняется с помощью отправки DSA двух различных операций DSP.

Если присутствует элемент **relatedEntry**, то должны применяться следующие специальные правила:

– при оценке подкомпонентов **infoTypes** компонента **selection** из **SearchArgument**, для **infoTypes** должно приниматься значение **attributeTypesAndValues** вне зависимости от первоначально установленного значения;

– все атрибуты, которые специфицированы в любом компоненте **joinAtt** для **JoinAttPair**, должны включаться в выбор, вне зависимости от того, были они или нет включены в него ранее;

– DSA, координирующий результаты для относящихся статей, должен опускать значения и не специфицированные аргументы, чтобы обеспечить соответствие результатов с первоначальным запросом пользователя.

Аргумент relatedEntry должен передаваться в следующем исходящем **ChainingArguments** тем DSA, который поддерживает относящиеся статьи.

v) Если граничащий DSA отличается от первоначального исполнителя (см. п. 15.5.5) и граничащий DSA поддерживает DSP постраничные результаты, то он может установить данный компонент равным **TRUE**, что сообщает первоначальному исполнителю о необходимости предоставить DSP постраничные результаты. Если этот компонент равен **FALSE** (по умолчанию), то первоначальный исполнитель не должен обеспечивать DSP постраничные результаты. Первоначальный исполнитель, который поддерживает DSP постраничные результаты, не должен передавать данный компонент тому (тем) DSA, которым он отправляет подзапросы.

w) Компонент **nonDapPdu** используется для индикации того, что PDU, который содержится в аргументе связывания, происходит не от DAP запроса, такого как запрос LDAP.

x) Компонент **streamedResults** используется в качестве счетчика для определения того, могут ли потоковые результаты быть связаны в ответ на данную операцию. Каждый DSA, который принимает участие в разрешении имени, увеличивает счетчик на единицу только в том случае, если счетчик присутствует, DSA понимает потоковые результаты и DSA готов принять потоковые результаты для данной связанной операции. Счетчик затем используется тем DSA, который завершает разрешение имени с целью определить – готов ли каждый из предыдущих DSA обрабатывать потоковые результаты.

y) Компонент **excludeWriteableCopies** важен только для операций Search и List; он сообщает о том, что поиск должен применяться к в первичным главным копиям статей, а не к копиям с возможностью записи для этих статей. Данный необязательный компонент может использоваться DSA в качестве одного из способов для того, чтобы избежать получения дублирующих результатов (см. п. 20.1).

10.4 Результаты связывания

Результаты **ChainingResults** получаются в результате выполнения каждой операции и обеспечивают обратную связь с DSA, который вызвал данную операцию.

```
ChainingResults::= SET {
        info                    [0]   DomainInfo OPTIONAL,
        crossReferences   [1]   SEQUENCE SIZE (1..MAX) OF CrossReference OPTIONAL,
        securityParameters      [2]   SecurityParameters DEFAULT { },
        alreadySearched   [3]   Exclusions OPTIONAL }
```

Различные компоненты имеют следующее значение:

a) Компонент **info** используется для передачи DMD-специфической информации между DSA, которые вовлечены в обработку общего запроса. Этот компонент имеет тип **DomainInfo**, который относится к типу без ограничений.

b) Компонент **crossReferences** присутствует в **ChainingResults** только тогда, когда в соответствующем запросе компонент **returnCrossRefs** имеет значение **TRUE**. Данный компонент состоит из последовательности элементов **CrossReference**, каждый из которых содержит **contextPrefix** и дескриптор **accessPoint** (см. п. 10.8).

```
CrossReference::= SET {
        contextPrefix       [0]   DistinguishedName,
        accessPoint         [1]   AccessPointInformation }
```

Перекрестная ссылка **CrossReference** может добавляться системным агентом DSA в том случае, когда обнаруживается сопоставление между частью аргумента **targetObject** операции и одним из ее контекстных префиксов. Административный орган DSA может располагать стратегией, которая не позволяет возвращать такое знание, и в данном случае элемент не будет добавлен к последовательности.

c) Тип данных **SecurityParameters** (параметры безопасности) специфицируется в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Отсутствие компонента **securityParameters** считается эквивалентным тому, что присутствует пустой набор параметров безопасности.

d) Компонент **alreadySearched** (если он присутствует) обозначает какие из имен RDN, подчиненных для данного **targetObject**, были обработаны в качестве части связанной операции Search (поиск) и, таким образом, должны быть исключены в последующем подзапросе.

ПРИМЕЧАНИЕ. – Названия в **contextPrefix** или **alreadySearched** должны являться первичными выделенными именами и не должны содержать альтернативные выделенные имена.

10.5 Прогресс операции

Значение **OperationProgress** описывает состояние продвижения при выполнении операции, в которой будут принимать участие несколько DSA.

```
OperationProgress::= SET {
        nameResolutionPhase     [0]     ENUMERATED {
            notStarted          (1),
            proceeding          (2),
            completed           (3) },
        nextRDNToBeResolved     [1]     INTEGER OPTIONAL }
```

Различные компоненты имеют следующее значение:

a) Компонент **nameResolutionPhase** обозначает, какая фаза операции достигнута при обработке имени **targetObject**. Там, где этот компонент сообщает для разрешения имени значение **notStarted**, тогда DSA еще не достиг контекста имени, который содержит начальный RDN имени. Если для разрешения имени указывается **proceeding**, тогда начальная часть имени была распознана, хотя содержащий целевой объект DSA пока не был достигнут. Значение **nextRDNToBeResolved** обозначает, какая часть имени была распознана [см. п. 10.5 b)]. Если разрешение имени указывается **completed**, то тогда был достигнут DSA, который содержит объект цели, и операция обрабатывается правильным образом.

b) Значение **nextRDNToBeResolved** сообщает DSA о том, какой из RDN в составе имени **targetObject** должен быть разрешен следующим. Это значение представляет целое число в диапазоне от единицы до количества RDN, содержащихся в имени. Данный компонент присутствует только в том случае, если компонент **nameResolutionPhase** имеет значение **proceeding**.

10.6 Трассировочная информация

Значение **TraceInformation** передает вперед запись о DSA, которые принимали участие в выполнении операции. Оно используется для обнаружения существования петель (а также для того, чтобы их избежать), которые могут возникать при несоответствии знания или при наличии петель псевдонимов в дереве DIT.

```
TraceInformation::= SEQUENCE OF TraceItem

TraceItem::= SET {
        dsa                 [0]  Name,
        targetObject        [1]  Name OPTIONAL,
        operationProgress   [2]  OperationProgress }
```

Каждый DSA, который участвует в продвижении операции к другому, добавляет новый пункт в конец последовательности **TraceItem**. Каждый такой пункт **TraceItem** содержит:

a) имя DSA, который производит добавление пункта;

b) имя **targetObject**, которое добавляющий новый пункт DSA получил во входящем запросе. Данный параметр опускается в том случае, если связываемый запрос поступил от DUA (в этом случае подразумеваемым значением является **object** или **baseObject** в **XOperation**), или же его значение равно значению (действительному или подразумеваемому) **targetObject** в **ChainingArgument** исходящего запроса;

c) **operationProgress**, которое добавляющий пункт DSA получил во входящем запросе.

dsa должен являться первичным выделенным именем и не должен содержать альтернативных выделенных имен. Каждый RDN в уже обработанном **targetObject** должен являться первичным RDN. Альтернативные выделенные имена с контекстом могут включать в компонент **valuesWithContext** из **AttributeTypeAndDistinguishedValue**, который содержится в RDN.

10.7 Тип ссылки

Значение **ReferenceType** определяет один из различных видов ссылок, которые определяются в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

```
ReferenceType::= ENUMERATED {
        superior                (1),
        subordinate             (2),
        cross                   (3),
        nonSpecificSubordinate  (4),
        supplier                (5),
        master                  (6),
        immediateSuperior       (7),
        self                    (8),
        ditBridge               (9) }
```

10.8 Информация о точке доступа

Существуют три типа точек доступа:

a) Значение **AccessPoint** определяет специальную точку, в которой может производиться доступ к Справочнику, а именно к DSA или серверу LDAP. При обращении к DSA, точка доступа должна иметь **Name** (имя), которое является именем данного DSA, а также может иметь **PresentationAddress**, который используется в OSI или IDM коммуникациях с данным DSA – в этом случае также не должен присутствовать labeledURI. При обращении к LDAP серверу точка доступа должна иметь **LabeledURI**, который используется в LDAP коммуникациях с этим сервером LDAP. Если присутствует **LabeledURI**, то **Name** и **PresentationAddress** должны игнорироваться, также не должен отсутствовать **SET OF protocolInformation**.

```
AccessPoint::= SET {
        ae-title            [0]  Name,
        address             [1]  PresentationAddress,
        protocolInformation [2]  SET SIZE (1..MAX) OF ProtocolInformation OPTIONAL,
        labeledURI          [6]  LabeledURI OPTIONAL }

LabeledURI::= DirectoryString{ub-labeledURI}
```

b) Значение **MasterOrShadowAccessPoint** определяет точку доступа к Справочнику. Категория **category** точки доступа, которая может принимать значения **master** или **shadow**, зависит от того, указывает она на контекст именования или же на повсеместно используемую дублируемую область. Компонент **chainingRequired** указывает, требуется ли связывание для данного DSA, например, что для данного DSA не должен возвращаться реферал.

```
MasterOrShadowAccessPoint::= SET {
    COMPONENTS OF              AccessPoint,
    category            [3]    ENUMERATED {
        master                 (0),
        shadow                 (1) } DEFAULT master,
    chainingRequired [5]   BOOLEAN DEFAULT FALSE }
```

c) Значение **MasterAndShadowAccessPoints** определяет набор точек доступа для данного Справочника, т. е. набор относящихся к нему DSA и/или серверов LDAP. Эти точки доступа имеют то общее свойство, что каждая ссылается на DSA или сервер LDAP, который содержит информацию о статьях в соответствии с общим контекстом именования (или же общего набора контекстов именования, собранных в одном DSA – если значение является значением атрибута **nonSpecificKnowledge**). Значение **MasterAndShadowAccessPoints** указывает на **category** для каждого значения **AccessPoint**, которую он содержит. Точка доступа главного DSA или сервера LDAP для контекста именования не должна включаться в данный набор.

ПРИМЕЧАНИЕ. – Разработчики должны определить, что для сервера LDAP возможно, даже если он определен как **shadow**, обновлять статьи в ответ на операцию обновления LDAP, которую он получает.

```
MasterAndShadowAccessPoints::= SET SIZE (1..MAX) OF MasterOrShadowAccessPoint
```

Значение **AccessPointInformation** указывает одну или более точек доступа для Справочника.

```
AccessPointInformation::= SET {
        COMPONENTS OF          MasterOrShadowAccessPoint,
        additionalPoints  [4]  MasterAndShadowAccessPoints OPTIONAL }
```

В том случае, если DSA принадлежит к первому изданию и он создает значение **AccessPointInformation**, то необязательный компонент набора отсутствует. Если DSA первого издания интерпретирует значение **AccessPointInformation**, то все имеющиеся значения **MasterAndShadowAccessPoints** будут проигнорированы.

Для DSA второго и последующих изданий значения компонента **MasterOrShadowAccessPoint**, который создается для значения **AccessPointInformation**, могут относиться к категории master или shadow, что определяется процедурой выбора знания того DSA, который создает это значение. Он может быть представлен как рекомендуемая точка доступа, предоставленная DSA, который создает значение для принимающего это значение DSA. Значение **MasterAndShadowAccessPoints** может быть также как опция создаваться для значения **AccessPointInformation**. Оно представляет дополнительную информацию, которая может использоваться в процедуре выбора знания принимающего DSA для определения альтернативной точки доступа.

10.9 Знание моста DIT

Значение **ditBridgeKnowledge** указывает на определенную точку, в которой может осуществляться доступ к другому DIT, в частности к DSA или серверу LDAP. **ditBridgeKnowledge** специфицирует точку доступа **accessPoint**, в которой можно получить доступ к данному DSA или серверу LDAP.

```
DitBridgeKnowledge  ::=  SEQUENCE {
        domainLocalID                 DirectoryString{ub-domainLocalID} OPTIONAL,
        accessPoints                  MasterAndShadowAccessPoints }
```

domainLocalID содержит читаемое описание, которое идентифицирует включенный в ссылку DIT.

10.10 Исключения

Как определено в п. 10.3, компонент **exclusions** для **ChainingArguments** используется с целью ограничить сферу операции Search (поиск). Он указывает определенное количество записей, подчиняющихся объекту цели, который (вместе со своими подчиненными) не должен включаться в обработку операцией Search (поиск). Компонент **exclusion** определяется как значение ASN.1 типа **Exclusions**.

```
Exclusions::= SET SIZE (1..MAX) OF RDNSequence
```

Каждое значение **RDNSequence** в наборе **Exclusions** должно идентифицировать контекстный префикс для контекста именования, подчиненного объекту цели. Если DSA получает запрос **search** со значением **RDNSequence**, которое не соответствует данному ограничению, то DSA может игнорировать это значение. Значение **RDNSequence** является связанным с целевым объектом и не является выделенным именем контекстного префикса.

Exclusions должны являться первичными выделенными именами. Также могут включаться альтернативные выделенные имена и информация о контексте.

Exclusions могут, кроме того как являться частью запроса пользователя, использоваться DSA для минимизации дублирования информации, возвращаемой подзапросами поиска при наличии теневой информации.

Рис. 5 иллюстрирует пример использования **Exclusions**. В данном примере DSA содержит две дублируемые области, одна из них располагается ниже другой. Одна из областей начинается с контекстного префикса X, вторая – с контекстного префикса C. Копия записи в Y имеет три подчиненные ссылки на контексты именования: A, B и C.

Если в качестве примера в данном DSA выполняется поиск в поддереве, начиная с базового объекта в контексте именования X, то DSA может обеспечить информацию из дублируемых областей X и C. Информация от контекстов именования A и B должна быть предоставлена с помощью подчиненных ссылок. При выполнении декомпозиции запроса ссылки продолжения, которые должны использоваться в **partialResults** или при связывании, специфицируют Y в качестве объекта цели, а C – в качестве одного из элементов набора **Exclusions**.

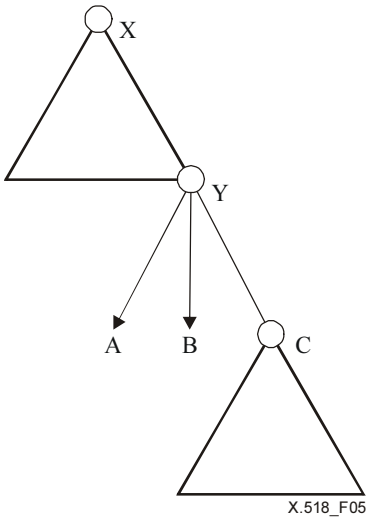


Рисунок 5 – Исключения

10.11 Ссылки продолжения

Ссылка продолжения **ContinuationReference** описывает, как выполнение всех или отдельных частей операции может быть продолжено другим DSA, сервером LDAP или их комбинацией. Обычно она возвращается как реферал, когда используемый DSA не может или не хочет самостоятельно распространять запрос далее.

```
ContinuationReference::= SET {
    targetObject          [0]  Name,
    aliasedRDNs           [1]  INTEGER OPTIONAL, -- только для систем первого издания
    operationProgress     [2]  OperationProgress,
    rdnsResolved          [3]  INTEGER OPTIONAL,
    referenceType         [4]  ReferenceType,
    accessPoints          [5]  SET OF AccessPointInformation,
    entryOnly             [6]  BOOLEAN DEFAULT FALSE,
    exclusions            [7]  Exclusions OPTIONAL,
    returnToDUA           [8]  BOOLEAN DEFAULT FALSE,
    nameResolveOnMaster   [9]  BOOLEAN DEFAULT FALSE }
```

Различные компоненты имеют следующее значение:

a) Компонент **targetObject** указывает имя, которое предлагается использовать при продолжении операции. Это имя может отличаться от того, которое было получено в **targetObject** входящего запроса в том случае, если, например, произошло разыменование псевдонима или же был обнаружен базовый объект поиска.

   Имена RDN в **targetObject** должны являться первичными RDN (для тех RDN, которые уже были обработаны). Также могут включаться альтернативные выделенные имена с контекстом.

b) Компонент **aliasedRDNs** указывает, сколько (при их наличии) из RDN в имени объекта цели были созданы при разыменовании псевдонима. Данный аргумент присутствует только в том случае, если осуществлялось разыменование псевдонима.

   ПРИМЕЧАНИЕ. – Данный компонент используется для совместимости с первым изданием реализаций Справочника. DUA (и DSA), которые реализуются в соответствии с более поздними изданиями спецификации Справочника, должны всегда опускать данный параметр из **CommonArguments** последующего запроса. Таким образом, Справочник не будет сигнализировать ошибку в том случае, если разыменование псевдонима распространяется на другие псевдонимы.

c) Значение **operationProgress** определяет достигнутый прогресс при разрешении имени, который будет управлять дальнейшим выполнением операции названными DSA, если получившие **ContinuationReference** DSA или DUA захотят ей следовать.

d) Значение компонента **rdnsResolved** (которое должно присутствовать только в том случае, если некоторые RDN в имени еще не являлись предметом полного разрешения имени, однако исходя из перекрестных ссылок допускается, что они являются правильными) указывает на то, сколько RDN было разрешено на самом деле с использованием исключительно внутренних ссылок.

e) Компонент **referenceType** указывает, какой тип знания использовался при генерации данного продолжения.

f) Компонент **accessPoints** указывает на точки доступа, с которыми необходимо контактировать для обеспечения данного продолжения. Только там, где используются неспецифические подчиненные ссылки, могут существовать несколько единиц **AccessPointInformation**.

g) Компонент **entryOnly** устанавливается равным **TRUE** в том случае, если первоначальной операцией являлся поиск, в котором аргумент **subset** был установлен как **oneLevel**, а статья псевдонима встречалась как непосредственный подчиненный **baseObject**. Тот DSA, который успешно выполнил разрешение имени для имени **targetObject**, будет выполнять оценку объекта только для названной записи.

h) Компонент **exclusions** идентифицирует набор подчиненных контекстов именования, которые не должны исследоваться принимающим DSA.

i) Элемент **returnToDUA** является необязательным и может поставляться тогда, когда создающий ссылку продолжения DSA желает указать на то, что он не хочет возвращать информацию через промежуточный DSA (например, по соображениям безопасности), а хочет сообщить, что информация доступна непосредственно с помощью операций над DAP или LDAP между инициировавшими запрос DUA или клиентом LDAP и данным DSA. Если **returnToDUA** устанавливается как **TRUE**, то **referenceType** может быть установлен как **self**.

j) Элемент **nameResolveOnMaster** является необязательным и может поставляться тогда, когда создающий ссылку продолжения DSA встречает NSSR. Если значение установлено как **TRUE**, то это говорит о том, что последующее разрешение имени, т. е. сравнение остающихся RDN из **nextRDNToBeResolved**, не должно использовать информацию из копий статей, включая записываемые копии в реализациях с многими мастерами; последующее разрешение каждого остающегося RDN будет производиться в главном DSA для статьи, которая идентифицируется данным RDN (см. п. 20.1).

11 Привязывание и развязывание

DSABind и **DSAUnbind**, соответственно, используются DSA в начале и конце периода доступа к другому DSA. Привязывание или развязывание ассоциации DSP не должно само по себе приводить к потере любых постраничных результатов, которые запрашивались в ходе данной ассоциации.

11.1 Привязывание DSA

Операция **DSABind** используется для того, чтобы начать период кооперации между двумя DSA, которые обеспечивают службы Справочника.

```
DSABind::= BIND
        ARGUMENT      DirectoryBindArgument
        RESULT        DirectoryBindResult
        BIND-ERROR    DirectoryBindError
```

Компоненты **DSABind** идентичны своим аналогам в DirectoryBind (см. Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3), однако имеются следующие различия:

– **Credentials** для **DirectoryBindArgument** позволяют переслать отвечающему DSA информацию, которая идентифицирует AE-заголовок инициирующего DSA. AE-заголовок должен иметь форму выделенного имени Справочника.

– **Credentials** для **DirectoryBindResult** позволяют переслать инициирующему DSA информацию, которая идентифицирует AE-заголовок отвечающего DSA. AE-заголовок должен иметь форму выделенного имени.

– Имя DSA или AE-заголовок могут использовать альтернативные выделенные имена и могут включать информационный контекст.

ПРИМЕЧАНИЕ 1. – Там, где имена используются либо в простых, либо в сильных рекомендациях, можно использовать альтернативные выделенные имена (если они существуют). Однако аутентификация и контроль доступа, которые основаны на использовании имени, могут не работать должным образом, если не используется первичное выделенное имя. Вслед за успешной обработкой аутентифицированной операции BIND, независимо от того, какое имя использовалось в качестве аргумента BIND, связанные предметы должны знать друг друга по их первичным выделенным именам, что облегчает проведение контроля доступа при сохранении эффекта привязывания BIND.

ПРИМЕЧАНИЕ 2. – Необходимые для аутентификации рекомендации могут переноситься элементами службы обмена безопасности Security Exchange Service Element (см. Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5), в этом случае они не присутствуют в аргументах связывания или в результатах.

11.2 Развязывание DSA

Прекращение связывания в конце периода кооперации между двумя DSA, которые обеспечивают службу Справочника, для окружения OSI специфицируется в пп. 7.6.4 и 7.6.5 Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5, а для окружения TCP/IP в п. 9.3.2 Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5.

12 Связанные операции

Для каждой из операций, которые используются для доступа к абстрактной службе Справочника, существует операция между сотрудничающими DSA, обеспечивающая взаимно-однозначное соответствие. Для того чтобы отразить это соответствие, используется префикс "Chained" (связанный), который ставится перед названиями операций, используемых между взаимодействующими DSA.

Аргументы, результаты и ошибки связанных операций являются, с единственным исключением, систематическим образом сформированными из аргументов, результатов и ошибок соответствующих операций в абстрактной службе Справочника (как описывается в п. 12.1). Единственным исключением является операция **ChainedAbandon**, которая синтаксически эквивалентна своему аналогу в службе Справочника (описывается в п. 12.2).

12.1 Связанные операции

DSA, который получает операцию от DUA или клиента LDAP, может выбрать построение связанной формы данной операции и передать ее далее другому DSA. DSA, который получил операцию в связанной форме, также может по своему выбору связать эту операцию с другим DSA. DSA, вызывающий связанную форму операции, может подписать, зашифровать, или же одновременно подписать и зашифровать аргумент этой операции; DSA, который выполняет операцию, если это было запрошено, может подписать, зашифровать, или же подписать и зашифровать результат или ошибку, которую возвратил отвечающий для данной операции. DSA, который получил операцию от клиента LDAP или же получил операцию LDAP от другого DSA, может по своему выбору передать далее серверу LDAP оригинальную операцию, переданную клиентом LDAP.

Связанная форма операции специфицируется с помощью параметризованного типа **chained { }**.

```
chained { OPERATION: operation } OPERATION::= {
        ARGUMENT   OPTIONALLY-PROTECTED {
            SET {
                chainedArgument       ChainingArguments,
                argument          [0] operation.&ArgumentType } }
        RESULT       OPTIONALLY-PROTECTED {
            SET {
                chainedResult         ChainingResults,
                result            [0] operation.&ResultType } }
        ERRORS { operation.&Errors EXCEPT referral | dsaReferral }
        CODE   operation.&operationCode }
```

ПРИМЕЧАНИЕ 1. – Операции абстрактной службы Справочника, которые могут использоваться в качестве действительных параметров для **chained { }**, включают ошибку **abandoned**. Присутствие данной ошибки среди набора возможных ошибок связанной операции отражает возможность того (обсуждается в п. 12.2), что **chainedAbandon** может быть сгенерирован для операции **chainedModify** при неисправности связанной ассоциации.

ПРИМЕЧАНИЕ 2. – Полная спецификация абстрактной службы DSA в Приложении A применяет этот параметризованный тип для создания всех связанных операций абстрактной службы.

Аргумент производной операции имеет следующие компоненты:

a) **chainedArgument** – это значение **ChainingArguments**, которое содержит информацию об аргументах, предоставленных первоначальными DUA или клиентами LDAP. Эта информация необходима для того, чтобы DSA или сервер LDAP могли выполнять свои операции. Данный тип информации определяется в п. 10.3.

b) **argument** – Это значение **operation.&Argument** и оно состоит из первоначального аргумента, предоставленного DUA (согласно спецификациям соответствующего раздела Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3), или же первоначального аргумента, предоставленного клиентом LDAP, как это специфицируется в соответствующем разделе RFC 2251.

   ПРИМЕЧАНИЕ 3. – Также возможно инкапсулировать типы PDU, отличающиеся от тех, которые происходят от DAP или LDAP, если это считается допустимым. Спецификация механизмов, которые позволяют сделать это, оставлены для последующего изучения.

Если запрос является успешным, то результат производной операции содержит следующие компоненты:

a) **chainedResult** – Это значение **ChainingResults** содержит информацию, которая дополнительно должна быть предоставлена инициировавшему DUA, и которая может являться необходимой для предшествующего DSA в цепочке. Этот тип информации определяется в п. 10.4.

b) **result** – Это значение **operation.&Result**, которое содержит результат, возвращенный исполнителем данной операции, и который должен быть возвращен как результат инициировавшему DUA. Данная информация специфицируется в соответствующем разделе Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3.

Если запрос завершается неудачно, то возвращается одна из ошибок, которая содержится в наборе **operation.&Errors**. Исключением является то, что вместо **referral** будет возвращаться **dsaReferral**. Набор ошибок, которые могут быть сообщены, соответствует описанию для соответствующих операций в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Ошибка **dsaReferral** описывается в п. 13.2.

12.2 Связанная операция прекращения

Операция **chainedAbandon** используется одним DSA для того, чтобы указать другому, что он более не заинтересован в выполнении запущенной ранее распределенной операции. Это может происходить по нескольким причинам, среди которых в качестве примера рассмотрим следующие:

- операция, которая привела к первоначальному связыванию между DSA, сама была прекращена или же была прекращена неявно при разрушении ассоциации;
- DSA получил необходимую информацию каким-то другим образом, например, от более быстро предоставившего ответ DSA, который участвовал в параллельном множественном связывании (parallel multi-chaining).

На DSA никогда не налагается обязательств по выпуску **chainedAbandon** или же обязательств действительно покинуть операцию при соответствующем запросе.

Если **chainedAbandon** удается действительно прекратить выполнение операции, то возвращается результат, а данная операция возвращает ошибку **abandoned**. Если **chainedAbandon** не удается прекратить операцию, то она самостоятельно возвращает ошибку **abandonFailed**.

12.3 Связанные операции и версия протокола

Операции, которые требуют версий протокола выше v1 (такие, как операция **modifyEntry** при использовании определенных аргументов) или которые возвращают другие результаты при использовании версии протокола выше v1 (такие, как **modifyEntry** с подписанным аргументом) должны связываться в ассоциации только с теми объектами, которые имеют выше или тот же номер версии по сравнению с объектами, которые использовались при отправке запроса.

13 Связанные ошибки

13.1 Введение

В большинстве случаев абстрактная служба DSA может возвращать те же ошибки, которые может возвращать абстрактная служба Справочника. К числу исключений относится то, что вместо **Referral** возвращается "ошибка" **dsaReferral** (см. п. 13.2), а следующие проблемы службы имеют одинаковый синтаксис, но различную семантику:

a) **invalidReference** – DSA, который возвращает данную ошибку, обнаружил ошибку в знании вызывающего DSA, что специфицируется в аргументе связывания **referenceType**.

b) **loopDetected** – DSA, который возвращает эту ошибку, обнаруживает петлю в информации о знании в данном Справочнике.

Порядок следования ошибок, которые могут происходить, определяется их порядком в абстрактной службе Справочника, согласно спецификациям Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3.

Если в процессе связанной операции происходит ошибка, то выполняющий ответ DSA может подписать, зашифровать, или же подписать и зашифровать возвращаемую ошибку.

13.2 Реферал DSA

Ошибка **dsaReferral** создается DSA тогда, когда по каким-то причинам он не желает продолжать операцию путем связывания с одним или несколькими другими DSA. Обстоятельства, при которых DSA может возвращать реферал, приводятся в п. 8.3.

```
dsaReferral   ERROR::= {
        PARAMETER OPTIONALLY-PROTECTED {
            SET {
                reference           [0]   ContinuationReference,
                contextPrefix       [1]   DistinguishedName OPTIONAL,
                COMPONENTS OF CommonResults } }
        CODE          id-errcode-dsaReferral }
```

Различные параметры имеют следующее значение:

a) **ContinuationReference** содержит информацию, которая необходима тому, кто вызвал эту операцию, чтобы передать далее соответствующий запрос – возможно, к другому DSA или серверу LDAP. Данный тип информации специфицируется в п. 10.11.

b) Если компонент **returnCrossRefs** в **ChainingArguments** для данной операции имеет значение **TRUE**, и реферал основывается на подчиненной или перекрестной ссылке, то как опция может также включаться параметр **contextPrefix**. Административный орган каждого DSA принимает решение о том, какие ссылки на знание, если они существуют, могут возвращаться подобным способом (другие, например, могут оставаться конфиденциальными для данного DSA).

Префикс **contextPrefix** или же Continuation Reference (ссылка продолжения) должны являться первичными выделенными именами. Альтернативные выделенные имена с контекстом могут включаться в компонент **valuesWithContext** из **AttributeTypeAndDistinguishedValue** для любого RDN.

Предоставленная информация может при желании также квалифицироваться с использованием компонента **notification** из **CommonResults**.

РАЗДЕЛ 5 – РАСПРЕДЕЛЕННЫЕ ПРОЦЕДУРЫ

14 Введение

14.1 Сфера применения и ограничения

Данный раздел специфицирует процедуры для распределенных операций Справочника, которые выполняются системным агентом Справочника DSA. Каждый DSA самостоятельно выполняет описанные ниже процедуры; коллективные действия всех DSA обеспечивают полный набор услуг, которые данный Справочник предоставляет пользователям.

14.2 Совместимость

Описание процедур DSA в данном разделе основывается на модели, которая представлена в пп. 8 и 9 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2 и пп. 7 и 8 данной спецификации Справочника. Диаграммы и соответствующее им текстовые описания являются средством отображения внешних (DAP, LDAP и/или DSP) входов DSA на один или более внешних выходов (т. е. результат, ошибка, реферал или связанный запрос), которые создаются данным DSA с учетом имеющегося дерева информации DSA, содержащегося в данном DSA.

Существует вероятность, что Справочник будет распределяться среди таких DSA, которые реализованы на основе различных изданий спецификации Справочника, а также тех, которые реализовывались для поддержки исключительно LDAP. Инициирующие запрос DUA или клиент LDAP не имеют представления о том, в соответствии с каким изданием были созданы один или несколько DSA, которые выполняют запрос DUA или клиентов LDAP. Итак, для того чтобы разрешить операции в подобном гетерогенном окружении, DSA должен быть реализован в соответствии с правилами расширяемости, которые определены в п. 12 Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5.

ПРИМЕЧАНИЕ 1. – SA, который реализован исключительно для поддержки LDAP, может быть реализован с учетом правил расширяемости или же может не учитывать эти правила.

Реализация DSA должна быть функционально эквивалентна внешнему поведению, которое специфицируется с помощью описываемых здесь процедур. Не стандартизируются алгоритмы, которые обеспечивают для данной реализации DSA получение корректного выхода(ов) для заданного входа(ов), а также не стандартизируется дерево информации DSA.

ПРИМЕЧАНИЕ 2. – Диаграммы, которые сопровождает процедуры, предназначаются для использования в качестве вспомогательных средств, поясняющие работу процедур. Они не должны рассматриваться как альтернатива текстовому описанию. Если для какой-то процедуры существует несоответствие между текстовым описанием и диаграммой, то считается, что приоритет должен отдаваться текстовому описанию.

14.2.1 Взаимодействие с участием DSA, относящегося к первому изданию

Если операция изменения выполняет оценку за границами DSA (например, **addEntry** с **TargetSystem**, Remove (удалить) или Rename (переименовать) контекстный префикс), то спецификация Справочника специфицирует только поведение двух DSA, издание которых относится ко второму или более позднему. Взаимодействие между двумя DSA первого издания, или же между DSA первого издания и DSA второго или более позднего издания, не относятся к области данной спецификации Справочника. Когда DSA с различными изданиями имеют иерархическое операционное связывание, то знание об издании другого DSA может позволить сообщить пользователю соответствующую информацию об ошибке.

14.3 Концептуальная модель

Сложность распределенной операции Справочника приводит к необходимости концептуальной модели, которая использует как описательные, так и иллюстративные методы. Однако используемые описательные или графические диаграммы не должны толковаться как формальное описание распределенных операций Справочника.

14.4 Индивидуальные и кооперативные операции DSA

Модель рассматривает операции DSA с использованием двух различных перспектив, которые при совместном использовании обеспечивают полную операционную картину Справочника.

a) **DSA-ориентированная перспектива** – В данной перспективе набор процедур, которые поддерживает Справочник, описываются с точки зрения единственного DSA. Это позволяет обеспечить определяющие требования для каждой процедуры и позволяет также полностью учесть их взаимоотношения и общую структуру управления. Пункты 16–22 описывают процедуры DSA с использованием перспективы, в которой центральное положение занимает DSA.

b) **операционно-ориентированная перспектива** – DSA-ориентированное представление обеспечивает полноту деталей, однако с ее использованием достаточно тяжело понять структуру отдельной операции, которая может подвергаться обработке несколькими DSA. Таким образом, в п. 15 рассматривается в основном операционно-ориентированное представление, которое позволяет представить каждую фазу в процессе обработки операции.

Для поддержки распределенных операций Справочника, каждый DSA должен выполнять действия, необходимые для реализации назначения каждой операции и дополнительных действий, которые необходимы для распределения данной реализации по многим DSA. Пункт 15 исследует различия между двумя этими методами. В пп. 16–22 оба этих типа рассматриваются подробно.

14.5 Кооперативные соглашения между DSA

Все DSA, которые используют отношения подчиненный/вышестоящий в соответствии с содержащимся в них контекстом именования, используют между собой иерархическое и/или неспецифическое иерархическое операционное связывание, которое определяется типами ссылок на знание, которое содержат эти DSA.

Иерархическое или неспецифическое операционное связывание между DSA может управляться с использованием процедур, определенных в пп. 24 и 25, или же с помощью других средств (например, телефона).

DSA, который содержит статьи, которые находятся внутри административной области вышестоящего DSA, должен администрировать подсхему, которая следует правилу управляющего поиска (при условии наличия) и должна контролировать доступ к статьям, как это требуется административным органом. Управление статьями в пределах административной области может выполняться согласно Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2 или же может выполняться местными механизмами.

15 Поведение распределенного Справочника

15.1 Кооперативное выполнение операций

Каждый DSA оснащен набором процедур, которые позволяют полностью обеспечить выполнение всех операций Справочника. В том случае, если DSA содержит DIB полностью, то все операции, фактически, полностью выполняются в пределах данного DSA. Если же DIB распределен среди многих DSA, то завершение типичной операции фрагментируется и лишь часть данной операции выполняется в каждом из потенциально большого количества взаимодействующих между собой DSA.

В распределенном окружении типичный DSA видит каждую операцию как переходное событие: операция вызывается DUA, клиентом LDAP или каким-то другим DSA, DSA выполняет обработку объекта и затем передает его другому DSA для последующей обработки.

Альтернативное представление рассматривает полную обработку, которая производится над операцией во время ее выполнения некоторым количеством взаимодействующими между собой DSA. Такая перспектива позволяет выявить одинаковые фазы в обработке, которые применимы ко всем операциям.

15.2 Фазы обработки операции

Каждая операция Справочника может быть представлена как состоящая из трех различных фаз:

a) фаза *Name Resolution* (*разрешение имени*), в которой имя объекта, со статьей которого должна производиться определенная операция, используется для определения DSA, который содержит данную статью;

b) фаза *Evaluation phase* (*фаза оценки*), в которой действительно выполняется операция, которая определена в данном запросе Справочника (например, операция Read (чтение));

c) фаза *Results Merging phase* (*фаза объединения результатов*), в которой результаты данной операции возвращаются DUA или клиенту LDAP, который их запрашивал. Если для взаимодействия выбран режим связывания, то фаза объединения результатов может затрагивать несколько DSA, каждый из которых участвовал в связывании первоначального запроса или подзапроса (как определяется в п. 15.3.1 – декомпозиция запроса) с другим DSA в течение одной или двух из предыдущих фаз.

В случае операций Read (чтение), Compare (сравнить), List (список), Search (поиск), Modify Entry (изменить статью), Modify DN (изменить DN) и Remove Entry (удалить статью), разрешение имени производится для имени объекта, которое передается в аргументе операции. В случае Add Entry (добавить статью) целевой статьей для разрешения имени является непосредственная вышестоящая статья для той, которая передается в аргументе операции – она может быть легко получена с помощью удаления последнего RDN из имени, которое передается аргументом операции. (Это делается с помощью локального аргумента **m** в процедуре **FindDSE**, п. 18.3.1.)

Операция над определенной статьей может первоначально направляться каждому DSA в Справочнике. Затем такой DSA использует свое знание, возможно в сочетании с другими DSA, для обработки операции с помощью указанных трех фаз.

15.2.1 Фаза разрешения имени

Name Resolution (разрешение имени) является процессом последовательного сопоставления каждого RDN в данном имени с дугой (или вершиной) в дереве DIT. Логически обработка начинается в корне (Root) и затем происходит в дереве DIT по нисходящей, сверху вниз. Однако так как дерево DIT может быть распределено между произвольным количеством DSA, то каждый DSA может выполнить только часть процесса разрешения имени. Данный DSA выполняет свою часть в процессе разрешения имени, перемещаясь по локальному информационному дереву DSA. Этот процесс описывается в п. 18, также прилагаются диаграммы (см. рис. 9–12). Используя локальное информационное дерево DSA и содержащуюся в нем информацию о знании, DSA

способен сделать заключение о том, может ли разрешение быть продолжено одним или несколькими DSA, или же имя является ошибочным.

Если установлена опция управления службой **manageDSAIT**, то фаза Name Resolution (разрешение имени) ограничивается работой в пределах информационного дерева DSA.

15.2.2 Фаза оценки

Когда завершается фаза разрешения имени, то затем действительно выполняется затребованная операция (например, Read (чтение) или Search (поиск)).

Операции, которые включают обращение к одной статье – Read (чтение) и Compare (сравнение) – могут выполняться полностью в пределах того DSA, в котором располагается данная статья.

Операции, в которых производится обращение ко многим статьям – List (список) или Search (поиск) – должны локализовать подчиненных цели, которые могут находиться или же не находиться в одном DSA. Если они не располагаются в одном DSA, то операция должна быть направлена тем DSA, которые специфицированы в подчиненной ссылке, неспецифической подчиненной ссылке, ссылке поставщика или главной ссылке (в зависимости от обстоятельств) с целью завершить процесс оценки.

Если установлена опция управления службой **manageDSAIT**, то фаза оценки ограничивается работой в пределах информационного дерева DSA. Аналогично, если фаза оценки запускается в пределах определенной административной области, то оценка ограничивается этой административной областью.

15.2.3 Фаза объединения результатов

Фаза объединения результатов начинается после того, когда становятся доступны какие-то результаты от фазы оценки.

В тех случаях, когда операция оказывает влияние только на одну статью, то результат операции может быть просто возвращен тому DUA или клиенту LDAP, который его запрашивал. В тех случаях, когда операция оказывает влияние на некоторое количество статей в нескольких DSA, результаты могут объединяться. Если для результатов реализуется защита, то не следует объединять результаты. Результаты будут возвращаться DUA или клиенту LDAP без выполнения слияния.

Разрешенные ответы, которые могут возвращаться заказчику после их слияния, включают следующие:

a) полный результат операции;

b) результат, который не является полным по той причине, что некоторые части дерева DIT остались неисследованными (применимо только к List (список) и Search (поиск)). Такой *частичный результат* может содержать ссылки продолжения для тех частей DIT, которые не были исследованы;

c) ошибка (специальным случаем является реферал); и

d) если заказчиком является DSA, может возвращаться **ChainingResults**.

15.3 Управление распределенными операциями

В аргумент каждой операции включается информация, которую DSA могут попросить выполнить и которая сообщают о выполнении каждой операции по мере того, как она проходит через различные DSA в данном Справочнике. Это позволяет для каждого DSA выполнить соответствующие аспекты необходимой обработки и записать завершение данного аспекта перед тем, как отправлять операцию наружу следующему DSA.

Дополнительные процедуры включаются в DSA для того, чтобы физически распределить операции и поддержать другие требования, которые возникают при их распределении.

15.3.1 Декомпозиция запроса

Декомпозиция запроса – это процесс, который выполняется DSA внутренним образом перед началом коммуникаций с одним или несколькими DSA или серверами LDAP. Запрос разбивается на несколько подзапросов, каждый из которых выполняет часть первоначальной задачи. Например, декомпозиция запроса может использоваться при операции поиска, после того, как был найден базовый объект. После проведения декомпозиции каждый из подзапросов может быть связан единичным или множественным образом с другими DSA и/или серверами LDAP с целью продолжить выполнение задачи.

Для связанного запроса (см. п. 12.1) или подзапроса значением **argument** должно являться неизмененный аргумент операции, если операция была инициирована DUA, и неизмененный LDAPMessage, если операция была инициирована клиентом LDAP. DSA, который получает связанный запрос, не должен изменять **argument** при осуществлении декомпозиции запроса.

ПРИМЕЧАНИЕ. – Следующие подразделы специфицируют требования к индивидуальным компонентам для **argument**. Это не должно интерпретироваться как то, что можно изменить компонент, который явно не упоминается.

15.3.2 DSA в качестве отвечающего на запрос

DSA, который получает запрос, может проверить ход выполнения этого запроса с помощью параметра **operationProgress**. Это помогает определить, находится ли операция в фазе Name Resolution (разрешения имени) или же достигла фазы оценки, и какую часть операции DSA должен попытаться выполнить. Если DSA не может полностью удовлетворить данный запрос, он должен либо передать (с помощью единичного или множественного связывания) операцию одному или нескольким DSA и/или серверам LDAP, которые могут

помочь при выполнении запроса, или же возвращает референал другому DSA или серверу LDAP, или же завершает запрос с ошибкой.

15.3.3 Завершение операции

Каждый DSA, который инициировал операцию или передавал эту операцию одному или нескольким другим DSA и/или серверам LDAP, должен отслеживать, существует ли данная операция до того момента, когда каждый DSA и/или сервер LDAP возвратит результат или ошибку, или же истекает максимальный лимит времени, отведенный для этой операции. Это требование применимо ко всем операциям, режимам передачи и фазам. Оно гарантирует корректное завершение распределенных операций, которые были переданы в данный Справочник.

15.4 Обработка петель

Дерево DIT может находиться в таком состоянии, при котором возможно образование петли. Например, петля может возникать при разрешении имени, когда разыменование одного или более псевдонимов приводит разрешение обратно к уже имевшемуся переходу в дереве DIT. Другой потенциальной причиной возникновения петель является неправильная конфигурация ссылок на знания.

В пределах контекста определенной операции Справочника, петля возникает каждый раз, когда операция возвращается к прежнему *состоянию*, при этом состояние определяется с помощью следующих компонентов:

- имя DSA, который осуществляет обработку операции в настоящее время;
- имя **targetObject**, которое содержится в аргументе операции;
- **operationProgress**, который содержится в аргументах операции и который определен в п. 10.5.

Это не означает, что не допускается многократная обработка операции одним DSA. Однако это означает, что DSA не может несколько раз обрабатывать одну операцию, которая находится в одном и том же состоянии.

Образование петли контролируется с использованием аргумента **traceInformation**, который был определен в п. 10.6, и который записывает последовательность состояний, через которые проходит данная операция. Для того чтобы определить возникновение или возможность возникновения петли, используются две стратегии. Это обнаружение петли и предотвращение петли, которые описываются в пп. 15.4.1 и 15.4.2, соответственно.

Обнаружение петли является обязательным, предупреждение петли обязательным не является.

15.4.1 Обнаружение петли

После получения операции Справочника DSA должен сначала проверить операцию с целью определить, может ли она прогрессировать. Важной задачей проверки является проверка петель, для этого определяют, встречалось ли текущее состояние операции в последовательности предыдущих состояний, которые сохраняются в аргументе **traceInformation** данной операции. Этот этап в проверке петли является обнаружением петли.

15.4.2 Предупреждение петли

Предотвращение петли требует, чтобы DSA непосредственно перед тем, как отправлять операцию другому DSA в качестве этапа процедуры связывания, определял – будет ли следующее состояние (которое представлено как **traceItem** и которое получивший операцию DSA добавит к **traceInformation** в тот момент, когда он ее получит) присутствовать в последовательности предыдущих состояний, которые хранятся в аргументе **traceInformation** для первоначальной входящей операции.

В тех случаях, когда получают референалы или производят действия с ними, предупреждение петли или обнаружение петли не может осуществляться только на основе проверки **traceInformation**. В этом случае, каждый раз, когда DSA выступает в качестве референала, он должен сохранить последовательные состояния операции (т. е. **traceItem** который принимающий DSA собирается добавить после получения входящего запроса). Перед тем, как осуществлять действия или возвращать референал, DSA должен провести проверку в этом списке с целью обнаружить, посылался ли уже аналогичный запрос при попытке обслужить входящую операцию.

15.5 Другие соображения при рассмотрении распределенных операций

15.5.1 Управление службой

Некоторые из возможностей по управлению службой заслуживают специального рассмотрения для распределенного окружения с целью обеспечить, чтобы операция обрабатывалась таким образом, как это было запрошено.

a) **chainingProhibited** – DSA консультирует данный управляющий элемент службы при определении метода распространения операции. Если это значение установлено, то DSA всегда использует режим референала. Если же оно не установлено, то DSA может выбирать между связыванием или референалом в зависимости от собственных возможностей.

b) **timeLimit** – DSA должен учитывать этот элемент управления чтобы удостовериться, что для данного DSA не превышен лимит времени. DSA, который DUA запрашивает выполнить эту операцию, сначала рассматривает **timeLimit**, который выражается DUA как время в секундах, которое остается до завершения операции. Если необходимо связывание, то **timeLimit** включается в аргумент связывания, который должен передаваться следующему(им) DSA. В этом случае для каждого запроса связывания используется одно и то же значение предела: это время (UTC), к

которому операция должна быть завершена, для того чтобы соответствовать первоначально установленным ограничениям. После получения **ChainingArguments** с указанным **timeLimit** получающий запрос DSA должен обеспечить соответствие данному пределу.

c) **sizeLimit** – DSA должен учитывать данный элемент управления с целью обеспечить, чтобы размер списка результатов не превысил указанный размер. Предел, который входит в общий аргумент первоначального запроса, передается при связывании запроса в неизменном виде. Если необходима декомпозиция запроса, то в аргумент, который должен передаваться следующему DSA, должно включаться одно и то же значение и для каждого подзапроса используется полное значение предела. При возвращении результатов запрашивающий DSA рассматривает множественные результаты и применяет предел к их общему количеству, что обеспечивает возвращение запрошенного количества результатов. Если предел был превышен, то это указывается в ответе.

d) **priority** – Во всех режимах распространения каждый из DSA отвечает за обеспечение того, что обработка операции упорядочивается в соответствии с поддержкой данного управления службой, если только оно присутствует.

e) **localScope** – Операция ограничивается локально определенной областью и ни один DSA не должен распространять запрос за эти пределы.

f) **scopeOfReferral** – Если DSA возвращает реферал или частичный результат для операций List (список) или Search (поиск), то встроенные ссылки продолжения должны находиться в пределах запрошенной области.

Все другие элементы управления также должны учитываться, однако их использование в распределенных условиях не требует каких-то специальных комментариев.

15.5.2 Расширения

Если DSA встречает расширенную операцию на стадии разрешения имени и определяет, что операция должна быть связана с одним или более DSA, то он должен включить в неизменном виде все присутствующие расширения.

ПРИМЕЧАНИЕ. – Административный орган может устанавливать, что следует возвращать **serviceError** с проблемой **unwillingToPerform** в том случае, если он не хочет передавать расширение.

Если DSA при обработке в фазе оценки встречает расширение, которое он не поддерживает, то могут возникать два возможных варианта. Если расширение не является критическим, то DSA должен игнорировать это расширение. Если же расширение является критическим, то DSA должен возвратить **serviceError** с проблемой **unavailableCriticalExtension**. Критическое расширение при операции со многими объектами может приводить к ошибкам подобного типа для результатов и для службы. DSA, который объединяет подобные результаты и ошибки, должен отбросить ошибки службы и использовать компонент **unavailableCriticalExtension** из **PartialOutcomeQualifier**, как это описывается в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3.

15.5.3 Разыменование псевдонима

Разыменование псевдонима представляет процесс создания нового имени для целевого объекта, которое производится с помощью замены статьи псевдонима части выделенного имени для оригинального имени целевого объекта на значение атрибута **AliasedEntryName** из статьи псевдонима. Имя **object** в данной операции при разыменовании псевдонима не затрагивается.

15.5.4 Разрешение имен, которые изменяются в зависимости от контекста

При разрешении имени по мере обработки RDN создается новое имя объекта цели, таким образом обеспечивается то, что каждое значение **AttributeTypeAndDistinguishedValue** в RDN использует первичное выделенное имя данного атрибута в качестве собственного значения **value**. Благодаря этому имя объекта цели прогрессирует по направлению к первичному выделенному имени. Это делается для того, чтобы обеспечить согласованную обработку названий, особенно тогда, когда в разрешении имени участвуют DSA, издание которых ниже третьего. Имя **object** в данной операции при проведении такой подстановки не затрагивается.

15.5.5 Постраничные результаты

Когда DUA включает **PagedResultsRequest** в запросах **search** или **list** (см. п. 7.9 в Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3), то разбиение на страницы может выполнять DSA, который непосредственно граничит с DUA (который также называется *граничащий DSA*) или же оно может выполняться DSA, который содержит статью **baseObject**/**object** для запроса **search** или **list** (возможно, после одного или более разыменования псевдонима), который также называется *начальный исполнитель*. Если разбиение на страницы выполняет граничащий DSA, который также может являться и начальным исполнителем, то разбиение на страницы носит название *результаты, разбитые на страницы граничащим DSA* (*bound-DSA paged results*). Если же разбиение на страницы выполняется начальным исполнителем, и этот начальный исполнитель не является граничащим DSA, то это называется *результаты, разбитые на страницы DSP* (*DSP paged results*).

Если DSA поддерживает результаты, разбитые на страницы DSP, то он обязан:

- поддерживать результаты, разбитые на страницы граничащим DSA (DSA-bound paged results);
- поддерживать результаты, разбитые на страницы DSP в качестве граничащего DSA;
- поддерживать результаты, разбитые на страницы DSP в качестве первоначального исполнителя, и
- поддерживать подкомпонент **entryCount** для **PartialOutcomeQualifier**.

Когда граничащий DSA получает запрос **search** или **list** с включенным параметром **PagedResultsRequest** и граничащий DSA не является начальным исполнителем данного запроса, то граничащий DSA может выбрать и включить параметр **dspPaging** в **ChainingArguments**. Начальный исполнитель может выбрать выполнение разбиения на страницы для результатов с помощью DSP. Это сигнализируется для граничащего DSA с помощью включения **queryReference** в **PartialOutcomeQualifier**. Это **queryReference**, которая возвращается пользовательскому агенту Справочника DUA с целью использования для получения следующей страницы.

Если начальный исполнитель либо не поддерживает разбиение результатов на страницы с помощью DSP, либо выбирает не выполнять это, то граничащий DSA может выполнить обычное разбиение на страницы граничащим DSA.

Если DSA является исполнителем, но не является начальным исполнителем, то он должен игнорировать возможный компонент **dspPaging** в **chainingArguments**, и должен также принимать во внимание управление службой **sizeLimit** (если она присутствует).

15.6 Аутентификация распределенных операций

Пользователи Справочника вместе с Административными органами, которые обеспечивают службы Справочника, могут по собственному усмотрению требовать аутентификации у операций Справочника. Для каждой операции Справочника сущность процесса аутентификации определяется используемой политикой безопасности.

Имеются два набора процедур аутентификации, которые позволяют при коллективном применении удовлетворить целому ряду требований к аутентификации. Первый набор процедур составляют те, которые обеспечиваются с помощью Bind (привязать): они облегчают проведение аутентификации между двумя прикладными объектами Справочника с целью установления ассоциации. Процедуры Bind обеспечивают широкий спектр обмена аутентификацией, начиная от простого обмена идентификаторами и заканчивая сильной аутентификацией.

В дополнение к аутентификации объектов на равных условиях для ассоциации, которую обеспечивает Bind, также определены дополнительные процедуры в пределах Справочника, которые обеспечивают аутентификацию для отдельных операций. Определены два различных набора процедур аутентификации Справочника. Один содействует службам аутентификации для инициатора, который использует аутентификацию с помощью DSA для инициатора первоначального запроса о службе. Второй набор содействует службам аутентификации результатов, которые направлены на аутентификацию, проводимую инициатором для всех возвращаемых результатов.

Для аутентификации инициатора определены две процедуры, первая из которых основывается на простом обмене идентификаторами и носит название **identity based authentication**, и вторая, которая основывается на технологиях цифровой подписи и носит название **signature based authentication**. Первая из этих процедур по своей сущности является рудиментарной, так как обмен идентификаторов основывается на обмене выделенными именами, которые передаются открытым способом.

Для аутентификации результатов определена единственная процедура **results authentication**, которая основывается на технологиях цифровой подписи. Благодаря достаточно сложному процессу сверки результатов для них не определяется более простая процедура, основанная на идентификаторах.

Также данные процедуры могут поддерживать аутентификацию ответных сообщений об ошибках.

Службы, которые описываются ниже, должны рассматриваться как дополнение к тем, которые обеспечивает служба Bind (привязывание); при этом считается, что процедуры Bind были успешно выполнены перед проведением аутентификации операций Справочника.

Процедуры, которые должны осуществляться DSA при проведении аутентификации инициатора и результатов, описываются в п. 22.

16 Диспетчер операций

В DSA основной контролирующей процедурой является **Operation Dispatcher** (диспетчер операций). Он проводит каждую операцию через все три этапа в обработке запроса. Таким образом, **Operation Dispatcher** использует набор процедур, который позволяют ему полностью обработать запрос (см. рис. 6).

16.1 Общие концепции

16.1.1 Процедуры

Каждая из процедур, которая используются диспетчером операций, состоит из определения концептуального интерфейса с помощью его параметров, т. е. аргументов, результатов и ошибок, а также описание самих этапов процедуры. Поведение процедур описывается с помощью диаграмм и текста. В диаграммах используемые символы имеют следующую семантику (см. рис. 7).

16.1.2 Использование общих структур данных

Все процедуры используют какие-то структуры данных, которые доступны при обработке операции в диспетчере операций (**Operation Dispatcher**). Эти структуры данных служат для координации потока данных в пределах **Operation Dispatcher**. Большинство из этих структур непосредственно связаны с аргументом операции и результатами, которые должны быть созданы данной операцией. Компоненты аргумента и результата обозначаются с использованием их имен, которые связаны с определениями ASN.1 (например, компонент

operationProgress для аргумента связывания). Если какие-то из этих структур являются составными структурами, то для обозначения компонента такой структуры может использоваться обозначение **compound.component** (например, **operationProgress.nameResolutionPhase**).

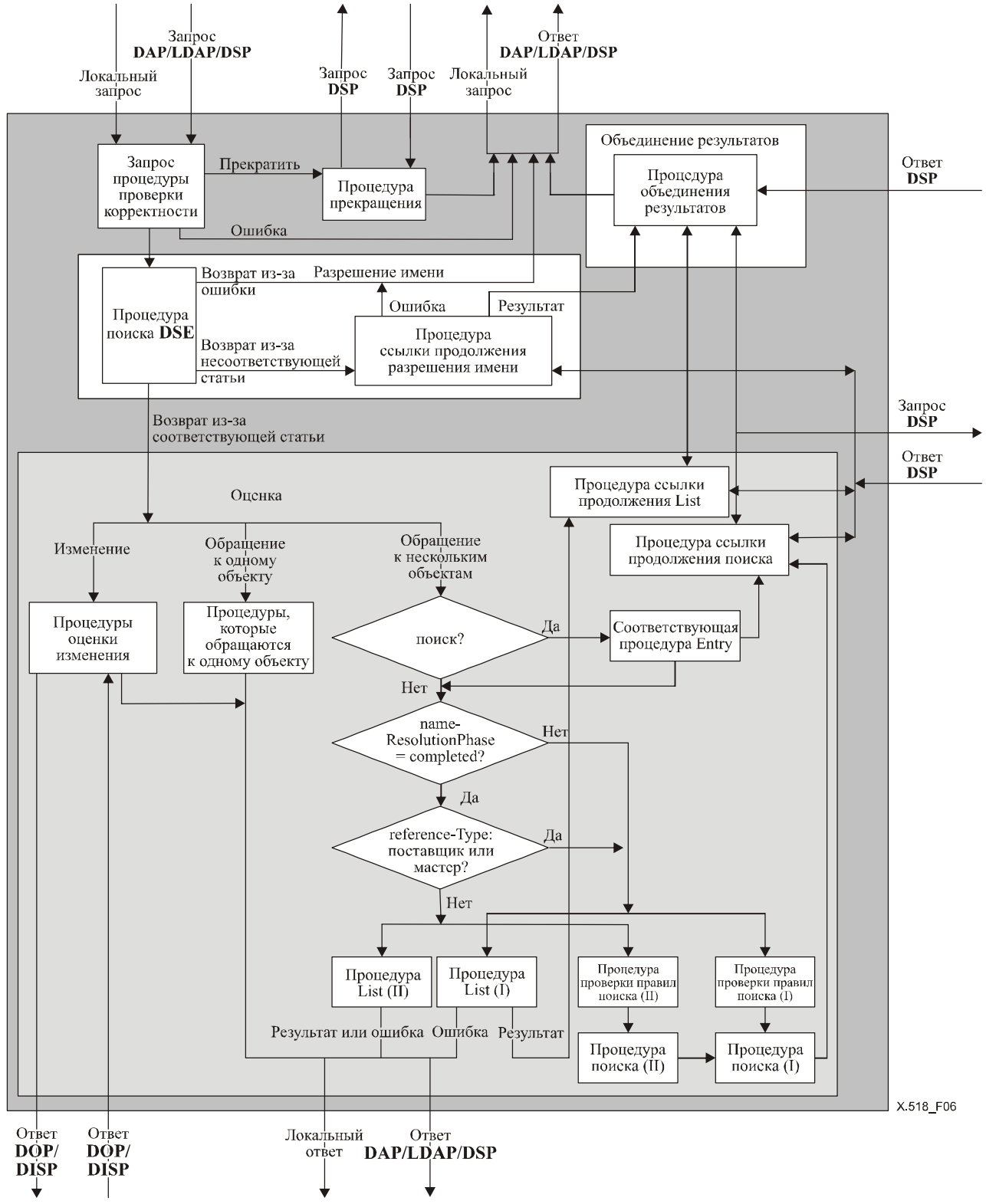


Рисунок 6 – Диспетчер операций

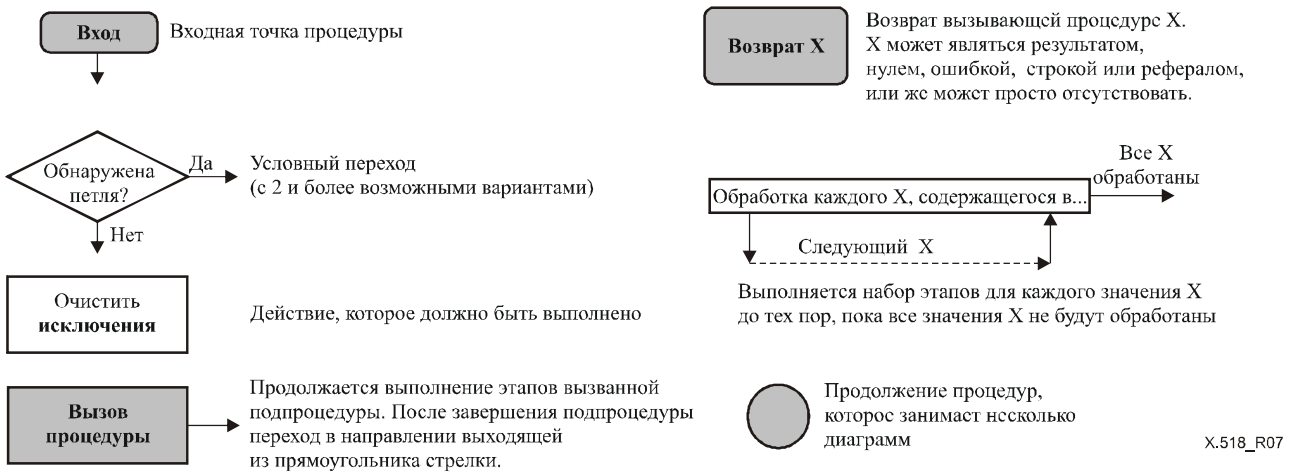


Рисунок 7 – Символы, которые используются в диаграммах

В диспетчере операций (**Operation Dispatcher**) определены следующие структуры данных:

- **NRcontinuationList** – Список ссылок продолжения, который создается для использования в процедуре **Name Resolution Continuation Reference**.
- **SRcontinuationList** – Список ссылок продолжения, который создается для использования в процедуре **List** или **Search Continuation Reference**.
- **admPoints** – Список ссылок на DSE с типом административная точка, которые собраны в процессе разрешения имени (Name Resolution).
- **referralRequests** – Список запросов или подзапросов, которые были связаны в результате выполнения рефералов. Каждый запрос/подзапрос представлен в форме **TraceItem**. Этот список используется процедурой предупреждения петель (Loop Avoidance), п. 15.4.2.
- **emptyHierarchySelect** – Переменная с типом Boolean, которая может устанавливаться в **Hierarchy Selection**. Переменная сбрасывается при первом входе в процедуру **Hierarchy Selection** при выполнении операции Search (поиск).
- **streamedResultsOK** – Переменная с типом Boolean, которая устанавливается в процедуре **Name Resolution** для индикации того, что для этой операции допускается использование потоковых результатов (streamed results). Для этой переменной значением по умолчанию является false (ложный).

Далее процедура может использовать набор определенных локально переменных.

16.1.3 Ошибки

На каждой стадии обработки можно обнаружить ошибку при выполнении какой-то подпроцедуры. Ошибка, которая обнаруживается в подпроцедуре, обычно возвращается заказчику как соответствующая ошибка протокола. В этом случае диспетчер операций немедленно завершает свою работу. В том случае, если получают большое количество ошибок, то локальные процедуры могут выбрать и возвратить одну из них.

В противном случае процедура может выбрать обработку ошибок (например, если возвращается **serviceError** с проблемой **busy** при связанном подзапросе поиска) на определенном этапе обработки операции. В этом случае процедура продолжает выполняться и заказчику не возвращается никаких ошибок.

По желанию DSA может подписывать, шифровать или же подписывать и шифровать ошибки, которые возвращаются распределенной операцией, основываясь на выбранном DirQOP и запрошенной защите ошибок.

16.1.4 Асинхронные события

Во время обработки запроса операции в диспетчере операций может произойти несколько асинхронных событий. В последующих параграфах мы обсудим, как обрабатывать превышение лимита времени, лимита размера или административного лимита, потерю ассоциации или же запрос Abandon (прекратить) для обрабатываемой в данный момент операции. Обработка всех других асинхронных событий, таких как решений локальной стратегии, выходит за рамки данной спецификации Справочника.

16.1.4.1 Лимит времени

Лимит времени (**timeLimit**), который специфицируется в **CommonArguments**, может закончиться во время выполнения операции. Обычно в таких случаях возвращается ошибка **serviceError** с проблемой **timeLimitExceeded** для того DUA, клиента LDAP или DSA, который выполнял запрос, и диспетчер операций

завершает свою работу. В противном случае процедура может по своему выбору обрабатывать данное событие каким-то другим способом (например, во время обработки запроса **search**).

Если DSA получает запрос от другого DSA и при этом превышен лимит времени, то он должен отправить ошибку **serviceError** с проблемой **timeLimitExceeded** и не должен производить какую-то последующую обработку запроса.

Если DSA имеет ожидающие обработки подзапросы и истекает лимит времени **timeLimit**, а к этому времени пока не получено никаких результатов, то следует возвратить запрашивающему устройству ошибку **serviceError** с проблемой **timeLimitExceeded**.

Если DSA имеет незавершенные подзапросы, уже существуют результаты и истекает **timeLimit**, то необходимо возвратить заказчику результат со следующим содержанием:

a) все собранные результаты до того момента, как истек лимит времени **timeLimit**;

b) компонент **limitProblem** результата-параметра **partialOutcomeQualifier** должен быть установлен как **timeLimitExceeded (Превышение лимита времени)**;

c) компонент **unexplored** результата-параметра **partialOutcomeQualifier** должен содержать значение ссылки продолжения для каждого набора DSA, которым были направлены подзапросы, но результаты которых не включены в отправляемые заказчику результаты, в дополнение к ссылкам продолжения для тех DSA, которым данный DSA не пытался отправить подзапросы.

16.1.4.2 Потеря ассоциации

При потере ассоциации с заказчиком утрачивается всякая возможность для возвращения результатов. В качестве опции DSA может для каждого незавершенного опрашивающего (под)запроса отправить запрос **chainedAbandon**, пока не будет потеряна ассоциация с данным DSA. Все ответы на подобные запросы **chainedAbandon** и все ответы на существующие (под)запросы должны быть отброшены. В случае результатов, для которых разбиение на страницы выполнено DSP, граничащий DSA должен прекратить все существующие страничные результаты и должен сгенерировать новый запрос постраничных результатов, выбрав для этого **abandonQuery** в **PagedResultsRequest**.

Если утрачивается ассоциация с одним из незавершенных связанных подзапросов, а ассоциация с заказчиком не потеряна, то DSA может, только в целях операции опроса, в качестве опции попробовать использовать альтернативную ссылку на другой DSA, который способен обработать связанный запрос (например, ссылку на теневой DSA после потери ассоциации с главным DSA). Если это не удается, то DSA должен действовать следующим образом:

1) Если для **operationProgress.nameResolution** установлено значение **notStarted** или **proceeding**, то необходимо возвратить заказчику либо ошибку **serviceError** с проблемой **unavailable** или же возвратить ошибку реферала, для которого ссылка продолжения содержит набор DSA, способных продолжить операцию. Если в фазе разрешения имени используются неспецифические подчиненные ссылки, и потеряны не все из рассматриваемых ассоциаций, то в качестве опции можно попытаться провести разрешение имени без использования тех DSA, с которыми потеряна ассоциация. Если это не удается, то возвращается либо ошибка **serviceError** с проблемой **unavailable**, или же ошибка реферала, которая содержит полный NSSR.

   Если DSA, используя локальное знание, знает (что может выражаться с помощью соответствующего значения **MasterOrShadowAccessPoint**), что необходимо связывание с DSA, с которым утеряна ассоциация, то он должен выбрать вариант с отправкой ошибки **serviceError** с проблемой **unavailable**, при этом компонент **notification** с типом данных **CommonResults** должен содержать:

   – атрибут извещения **dSAProblem** со значением **id-pr-targetDsaUnavailable**; а также

   – атрибут **distinguishedName**, который в качестве значения использует выделенное имя данного DSA.

2) Если **operationProgress.nameResolution** установлен как **completed** и запрос представляет собой операцию над одним объектом, то заказчику возвращается ошибка **serviceError** с проблемой **unavailable**.

3) Если **operationProgress.nameResolution** установлен как **completed** и запрос представляет собой операцию над несколькими статьями, то DSA должен добавить ссылку продолжения на **partialOutcomeQualifier.unexplored** результата операции, где **AccessPointInformation** определяет набор DSA, которые могут продолжать операцию, включая те DSA, ассоциация с которыми потеряна.

16.1.4.3 Прекращение операции

В процессе обработки операции может быть получен запрос Abandon (прекращение) для данной операции. В этом случае при обработке запроса прекращения для операции, которая должна быть прекращена, вызывается процедура **Abandon**.

16.1.4.4 Административные лимиты

Могут существовать пределы, которые устанавливаются локальным администратором DSA или же самой реализацией DSA, например, такие пределы могут ограничивать время обработки запроса или же максимальный размер возвращаемых данных и так далее. Если превышены какие-то из этих пределов, DSA должен возвращать ошибку **serviceError** с проблемой **administrativeLimitExceeded** или же возвратить частичный результат (который собирается из набора уже имеющихся результатов) с ошибкой **limitProblem**, установленной как **administrativeLimitExceeded** (превышение административного лимита).

Дополнительная информация возвращается в атрибуте извещения **dSAProblem** следующим образом:

a) если лимит определен Администратором, то атрибут извещения **dSAProblem** должен принимать значение **id-pr-administratorImposedLimit**;

ПРИМЕЧАНИЕ. – Это не означает, что реализация должна предусматривать специальные возможности для администратора по установке административных лимитов.

b) если лимит вызван ограничениями реализации и проблема носит постоянный характер, атрибут извещения **dSAProblem** должен принимать значение **id-pr-permanentRestriction**;

c) если лимит вызван ограничениями реализациями и проблема не носит постоянного характера, например, вызывается временным переполнением, то атрибут извещения **dSAProblem** должен принимать значение **id-pr-temporaryRestriction**.

16.1.4.5 Лимит размера

Лимит размера, который определяется в **CommonArguments**, может быть превышен в любой момент при обработке операций List (список) или Search (поиск). В этом случае запрашивающему возвращается частичный результат (который получается из набора уже имеющихся результатов), где **limitProblem** установлен как **sizeLimitExceeded**. В дополнение к этому, можно использовать компонент **unexplored** для возврата ссылки продолжения для тех DSA, к которым доступ не производился.

Если операция является операцией поиска и установлена опция управления поиском **entryCount**, то DSA должен выполнить как можно лучшую оценку того, как много записей потенциально может быть возвращено, если бы не существовал лимит размера, при этом учитывается контроль доступа и не учитывается иерархический выбор. Затем DSA возвращает это значение в компоненте **entryCount** из **PartialOutcomeQualifier** с использованием варианта **bestEstimate** для того случая, когда не существует **unaccessed** DSA (к которым не производились обращения), в противном случае выбирается вариант **lowEstimate**.

После этого работа диспетчера операций завершается.

16.2 Процедуры диспетчера операций

Процедура, которая выполняется **Operation Dispatcher** при обработке каждого полученного запроса (с помощью DAP, LDAP или DSP), определяется с помощью следующих этапов. В результате разыменования псевдонимов данная процедура может также вызывать саму себя (локальный запрос), в этом случае возвращается локальный ответ (а не ответ от DAP, LDAP или DSP).

1) Проверяется несколько аспектов аргументов операции (процедура **Request Validation**). Если в процессе проверки обнаруживается ошибка, то эта ошибка возвращается локально или с помощью DAP/LDAP/DSP.

2) Если полученная операция представляет собой операцию Abandon (прекращение), то вызовите процедуру **Abandon** и после этого возвратите ответ.

3) Выполняется разрешение имени целевого объекта помощью выполнение процедуры **Find DSE** (которая включает в себя подпроцедуры **Target Found** и **Target Not Found**). Если запрошенная статья найдена и она соответствует требованиям (в соответствии с настройками управления службой, аргументов связывания и решений локальной стратегии), то переходим в фазу оценки **Evaluation Phase**, этап 6). Если в процессе разрешения имени встречается ошибка, то возвращается эта ошибка. Если запись не соответствует требованиям, то переходим к этапу 4).

4) Процедура **Name Resolution Continuation Reference** вызывается для обработки списка ссылок продолжения, который хранится в **NRcontinuationList**. Для того чтобы обработать эти ссылки продолжения, могут отправляться связанные запросы другим DSA (если управление службой и локальная стратегия позволяют сделать это).

   В случае возникновения ошибки эта ошибка непосредственно возвращается либо локально, либо с помощью DAP/LDAP/DSP. Если связанный запрос создает результат, то переходим к этапу 5).

5) Процедура **Result Merging** вызывается для объединения локальных результатов с полученными связанными результатами. Если связанные результаты содержат встроенные ссылки продолжения, то они могут быть разрешены, если это требуют или разрешают управление службой и локальная стратегия.

   В результате это может приводить к тому, что будут выпущены дополнительные связанные запросы (и эти связанные запросы также могут содержать встроенные ссылки продолжения).

   Объединенные результаты возвращаются тому, кто их запрашивал, после чего обработка запроса прекращается.

   Если для результатов выполняется защита, то объединение результатов не выполняется.

6) Если операция является операцией изменения, то переходим к этапу 7).

   Если операция представляет операцию обращения к единственной статье, то переходим к этапу 8).

   Если операция представляет операцию обращения ко многим статьям, то переходим к этапу 9).

7) При выполнении процедуры изменения может понадобиться образовывать, изменять или уничтожать операционное связывание (Operational Bindings), или же может понадобиться обновить теневое копирования как следствие выполнения операции. То, выполняется ли это

синхронно или асинхронно с выполнением первоначальной операции, зависит от соответствующих операций изменения (и также от локальной стратегии). Вызывающей стороне возвращается локальный результат или ошибка или же результат или ошибка DAP/LDAP/DSP.

8) Результат операции, которая обращается к единственной статье, непосредственно возвращается вызывающей стороне как локальный результат или результат DAP/LDAP/DSP.

9) Если операция является операцией обращения ко многим статьям, то необходимо проверить значение **nameResolutionPhase** для данной операции. Если значение не равно **completed**, то вызываются процедуры **List(I)** или **Search(I)**, в противном случае вызываются соответственно процедуры **List(II)** или **Search(II)**.

10) Результаты вызова процедуры **List(II)** (результат или ошибка) и результаты вызова процедуры **List(I)** (если в результате была получена ошибка) могут возвращаться непосредственно вызывающей стороне (как локальный результат или результат DAP/LDAP/DSP).

    Если процедура вызывает процедуру **List(I)**, то результат может содержать ссылки продолжения, которые уже были разыменованы (в зависимости от управления службой и локальной стратегии). Это может приводить к тому, что связанные операции List будут переданы не тем DSA, которым они предназначены. Для того чтобы объединить эти результаты, переходим к этапу 5) и вызываем процедуру **Result Merging**.

11) Если операция являлась операцией Search (поиск), то все ссылки продолжения разрешаются с помощью процедуры **Search Continuation Reference** (если это требуется и является допустимым). Это может привести к тому, что запросы связанных операций поиска могут быть переданы не тем DSA, которым они предназначены. Процедура **Result Merging** [см. этап 5)] вызывается для объединения результатов поиска и, возможно, для разыменования содержащихся ссылок продолжения (если они имеются).

16.3 Обзор процедур

Данный раздел представляет собой обзор основных возможностей процедур, которые используются диспетчером операций **Operation Dispatcher** и которые были определены в пп. 17–22.

16.3.1 Процедура Request Validation (запрос проверки корректности)

Данная процедура, которая описывалась в п. 17, вызывается для проверки петель, проверки лимитов, а также проверки безопасности перед выполнением локального разрешения имени. Также данная процедура обеспечивает значения по умолчанию для тех параметров **ChainingArgument**, которые не обеспечиваются DAP или LDAP в том случае, если запрос поступил от DUA или от клиента LDAP. Далее, эта процедура выделяет любой запрос **abandon** и извещает об этом **Operation Dispatcher**.

16.3.2 Процедура прекращения

Данная процедура (описывается в п. 20.5) пытается найти операцию, которая должна быть прекращена и затем завершает выполнение этой операции. Если существуют ожидающие обработки подзапросы, то связанное прекращение может посылаться после них. Процедура либо возвращает запрашивающей стороне пустой результат, либо сообщает об ошибке (например, ошибка **abandonError** с проблемой **tooLate**).

16.3.3 Процедура поиска DSE

Данная процедура (описывается в пп. 18.2–18.3) проводит сопоставление компонента имени целевого объекта и DSE, которые хранятся локально, с целью разрешить имя целевого объекта. Если встречается DSE псевдонима, то псевдоним разыменовывается (если это разрешается) и процедура запускается вновь с целью выполнить разрешение нового имени.

Если цель не была найдена, то процедура продолжается в подпроцедуре **Target Not Found**. Если цель найдена, то процедура продолжается в подпроцедуре **Target Found**.

ПРИМЕЧАНИЕ. – **Target Not Found** и **Target Found** являются продолжениями процедуры **Find DSE**.

Процедура может приводить к различным ошибкам, в этом случае запрашивающей стороне возвращается соответствующая ошибка протокола и выполнение **Operation Dispatcher** прекращается.

16.3.3.1 Подпроцедура Target Not Found (цель не найдена)

Данная процедура (описывается в п. 18.3.2) выполняет оценку для локализованных промежуточных DSE и создает набор ссылок продолжения в **NRcontinuationList**, который основывается на ссылках на знания, которые были обнаружены во время процедуры **Find DSE**. Данный набор ссылок затем обрабатывается в процедуре **Name Resolution Continuation Reference**.

Процедура может приводить к различным ошибкам, в этом случае запрашивающей стороне возвращается соответствующая ошибка и диспетчер операций **Operation Dispatcher** прекращает свою работу.

16.3.3.2 Подпроцедура Target Found (цель найдена)

Данная процедура (определяется в п. 18.3.3) проверяет, соответствует ли обнаруженный DSE требованиям запрашиваемой операции, а именно при использовании теневой информации. Это может включать в себя проверку соответствия для всего поддерева теневой информации ниже целевого объекта в том случае, если операция производится со многими объектами (например, поиск по поддереву).

Если обнаруженная запись соответствует требованиям, то вызывается процедура оценки для соответствующей операции. В противном случае в **NrcontinuationList** создается ссылка продолжения **ContinuationReference**, которая указывает на поставщика (или мастера) данной информации, и вызывается процедура **Name Resolution Continuation Reference**.

16.3.4 Процедура, которая обращается к единственной записи

Данная процедура (описывается в п. 19.2) вызывается для действительного исполнения операций, которые затрагивают единственную запись, т. е. это операции Read (чтение) и Compare (сравнение). После их завершения запрашивающей стороне (клиент DSA/DUA/LDAP) возвращается созданный данной процедурой ответ (результат или ошибка).

16.3.5 Процедуры изменения

Данные процедуры (описываются в п. 19.1) выполняются для обработки операций изменения, таких как Add Entry (добавить статью), Remove Entry (удалить статью), Modify Entry (изменить статью) и Modify DN (изменить DN). Это достигается с помощью выполнения специфических подпроцедур, которые определяются для каждой из этих операций. Во время (или после) выполнения этих подпроцедур другим DSA могут отправляться запросы DOP и DISP. После успешного завершения результат (который создается подпроцедурой) возвращается к запрашивающей стороне – клиенту DSA/DUA/LDAP.

16.3.6 Процедуры, которые обращаются ко многим записям

Данные процедуры (описываются в п. 19.3) выполняются для обработки операций, затрагивающих несколько записей, которые могут находиться или не находиться в пределах одного DSA. Это осуществляется с помощью специфических подпроцедур, которые определяются для каждой из операций Search (поиск) и List (список) с целью завершения декомпозиции запроса. Данные процедуры создают локальный результат для операции оценки, а также могут создавать набор ссылок продолжения в **SRcontinuationList**. Если список **SRcontinuationList** при завершении процедуры оказывается пуст, то созданный результат непосредственно возвращается к запрашивающей стороне в лице клиента DSA/DUA/LDAP. Если это операция Search (поиск), если результат является пустым и если установлено **emptyHierarchySelect**, тогда необходимо возвратить в компоненте **notification** из **PartialOutcomeQualifier** следующее:

– атрибут извещения **searchServiceProblem** со значением **id-pr-emptyHierarchySelection**.

Если список **SRcontinuationList** не пуст, то ссылки продолжения обрабатываются с помощью вызова процедур **List** или **Search Continuation Reference**, которые вызываются в зависимости от типа операции.

16.3.7 Процедура ссылки продолжения для разрешения имени

Данная процедура (описывается в п. 20.4.1) обрабатывает ссылки продолжения **NRcontinuationList**, которые были созданы в фазе разрешения имени. Эти ссылки продолжения либо используются для отправки связанных подзапросов, либо возвращаются в ошибке реферала. В случае связывания результаты или ошибки, которые были получены от связанного запроса, возвращаются для последующей обработки с помощью процедуры **Result Merging**.

16.3.8 Процедура ссылок продолжения операций List и Search

Эти процедуры (описываются в пп. 20.4.2 и 20.4.3) обрабатывают ссылки продолжения в **SRcontinuationList**, которые были созданы процедурами, обращающимися ко многим записям, и либо выполняют их разрешение с помощью отправки связанных подзапросов, либо выполняют их разрешение с помощью ссылки(ок) продолжения в пределах **partialOutcomeQualifier.unexplored**. Когда поступают результаты или ошибки для всех сделанных запросов, то они возвращаются для дальнейшей обработки с помощью процедуры **Result Merging**.

16.3.9 Процедура объединения результатов

Данная процедура (описывается в п. 21) либо проверяет результат связанного запроса, либо объединяет результаты локальной операции с результатами, полученными от связанных подзапросов. Если подзапрос возвратил ошибку, то процедура определяет, как должна обрабатываться данная ошибка.

Если в результате остаются какие-то ссылки продолжения, то они (если это разрешает локальная стратегия и требуется управлением службой) могут быть разыменованы процедурами **Name Resolution**, **List** или **Search Continuation Reference**, соответственно. Дублирующие результаты удаляются, если только результаты не являются подписанными.

Объединенный результат (включая все объединенные результаты и неразрешенные ссылки) возвращается запрашивающей стороне – DUA/клиенту LDAP/DSA.

Если для результатов осуществляется защита, то объединение результатов не должно выполняться.

17 Процедура Request Validation (проверка корректности запроса)

17.1 Введение

Процедура **Request Validation** является точкой входа диспетчера операций для входов от DUA, клиентов LDAP и DSA, которая осуществляет подготовку таких входов к обработке разрешения имени. Функцией этой

процедуры является обнаружение операции прекращения, выполнение проверки безопасности, подстройка входов, полученных от DUA или клиентов LDAP с целью позволить им проходить обработку таким же образом, как и входов от DSA, проверка аргументов запроса на предмет правильности синтаксиса и семантики, выполнение проверки петель и выполнение других разнообразных проверок. Диаграмма выполнения процедуры **Request Validation** представлена на рис. 8.

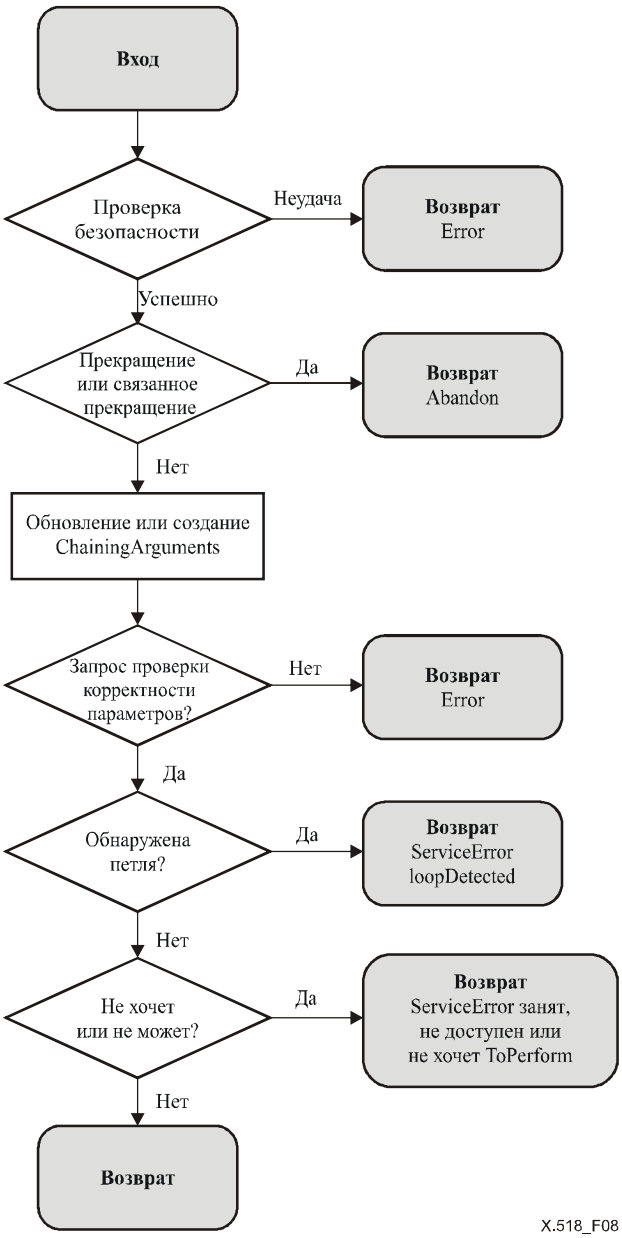


Рисунок 8 – Процедура Request Validation

17.2 Параметры процедуры

17.2.1 Аргументы

Если запрос поступает от DSA и аргумент отправляется создателем данного запроса, то входной аргумент для **Request Validation** состоит из **ChainingArguments** (за исключением операции **chainedAbandon**).

17.2.2 Результаты

Выходной результат **Request Validation** может принадлежать к одному из пяти возможных вариантов.

a) Если проверка безопасности завершается неудачей, то запрашивающей стороне возвращается ошибка.

b) Если в качестве входа используются операции **abandon** или **chainedAbandon**, то на выходе появляется аргумент операции.

c) Если аргумент запроса является некорректным, то запрашивающей стороне возвращается ошибка. В зависимости от локальной стратегии DSA может выбрать возвратить ли ошибку **serviceError** или же возвратить ошибку **securityError**.

d) Если обнаруживается петля, то запрашивающей стороне возвращается ошибка **serviceError** с проблемой **loopDetected**.

e) Если, по причине проблем с ресурсами или исходя из стратегии, DSA не может или не хочет выполнять операцию, то запрашивающей стороне возвращается ошибка **serviceError** (с проблемой **busy**, **unavailable** или **unwillingToPerform**). Если это уместно, то может возвращаться **serviceError** с проблемой **dataSourceUnavailable**.

f) Для всех других случаев проверенное на корректность входное значение, которое в случае поступления от DUA или клиента LDAP преобразуется с помощью добавления **ChainingArguments**, а в случае поступления от DSA приводит к обновлению **ChainingArguments.traceInformation**, поступает на выход процедуры и затем последовательно на вход процедуры **Name Resolution**.

17.3 Определение процедуры

Выполняется проверка безопасности, которая описывается в п. 17.3.2. В результате этого может быть возвращена ошибка, которая приводит к завершению работы диспетчера операций (Operation Dispatcher).

Если на входе присутствует операция **abandon** или **chainedAbandon**, то последовательно выполняются только этапы, представленные в п. 17.3.1, в противном случае выполняются этапы из пп. 17.3.3–17.3.5. Пункт 17.3.5 описывает процедуру определения петли, которая может приводить к возврату ошибки и завершению работы диспетчера операций.

Далее выполняются проверки в соответствии с п. 17.3.6. Они могут возвращать ошибку и приводить к завершению работы диспетчера операций.

Если же проверки в пп. 17.3.2–17.3.6 не приводят к завершению работы диспетчера операций, то выполняются этапы согласно п. 17.3.7 и процедура завершает работу, передавая выходные значения процедуре **Name Resolution**.

17.3.1 Обработка прекращения

Аргумент **abandon** или **chainedAbandon** передается процедуре **Abandon** (см. п. 20.5) для обработки запроса о прекращении.

17.3.2 Проверки безопасности

Если аргумент к данной операции подписан, зашифрован или же подписан и зашифрован, то можно проверить соответствующую подпись. Если подпись неверна или же ее расшифровка заканчивается неудачей, или же подпись отсутствует там, где она должна присутствовать, то запрашивающей стороне может возвращаться ошибка. В противном случае DSA может выполнять любое другое действие, которое определено локально.

17.3.3 Подготовка входа

17.3.3.1 Запрос DUA или клиента LDAP

Если операция получена от DUA или клиента LDAP, то значение **ChainingArguments** создается следующим образом:

a) **ChainingArguments.originator** устанавливается так, как описывается в п. 10.3.

b) **ChainingArguments.operationProgress** устанавливается равным значению **CommonArguments.operationProgress**.

c) **ChainingArguments.traceInformation** устанавливается как последовательность, в которой содержится единственное значение **TraceItem**. Данное значение конструируется следующим образом. **TraceItem.dsa** устанавливается равным имени DSA, который выполняет **Request Validation**. **TraceItem.targetObject** должен быть опущен. **TraceItem.operationProgress** устанавливается равным входящему значению.

d) Если управление службой для операции специфицирует лимит времени (время в секундах, которое остается для завершения операции), то **ChainingArguments.timeLimit** устанавливается равным (UTC) времени, к которому должна быть завершена операция, для того чтобы удовлетворить лимиту времени, который был определен пользователем.

e) **ChainingArguments.AuthenticationLevel** и **ChainingArguments.UniqueIdentifier** устанавливаются в соответствии с локальной стратегией безопасности.

f) **ChainingArguments.nameResolveOnMaster** копируется из **CommonArguments.nameResolveOnMaster**.

g) **ChainingArguments.exclusions**, **ChainingArguments.entryOnly** и **ChainingArguments.referenceType** копируются из **CommonArguments.exclusions**, **CommonArguments.entryOnly** и **CommonArguments.referenceType** если они присутствуют, в противному случае они опускаются.

h) Если в **ServiceControls** установлена опция **manageDSAIT**, тогда:
   – компонент **nameResolutionPhase** из **operationProgress** должен быть установлен как **completed**;

- компонент **nextRDNToBeResolved** из **operationProgress** опускается;
- **referenceType** должно принимать значение **self**;
- **entryOnly** должно принимать значение **FALSE**;
- **nameResolveOnMaster** должно принимать значение **FALSE**; и
- опция **chainingProhibited** из **ServiceControls** должна быть установлена;
- остающиеся необязательные элементы из **ChainingArguments** опускаются, там, где это специфицируется, допускаются значения по умолчанию.

i) Если не установлена опция **manageDSAIT** из **ServiceControls**, то опускаются остающиеся необязательные элементы из **ChainingArguments** и там, где это специфицируется, они принимают значения по умолчанию.

j) **ChainingArguments.SecurityParameters.ProtectionRequest** используется для индикации уровня защиты (подпись, шифрование, подпись и шифрование), который должен применяться для результатов.

17.3.3.2 Запрос LDAP

Если операция была получена от клиента LDAP, то значение **ChainingArguments** создается согласно п. 17.3.3.1, за тем исключением, что значение **ChainingArguments.operationProgress** должно быть установлено как **nameResolutionPhase notStarted**, а значения для **ChainingArguments.exclusions**, **ChainingArguments.entryOnly** и **ChainingArguments.referenceType** должны быть опущены.

17.3.3.3 Запрос DSA

Если операция была получена от DSA, то **ChainingArguments.traceInformation** обновляется с помощью добавления значения в конец последовательности **TraceItem**. Это значение создается следующим образом:

a) **TraceItem.dsa** устанавливается равным имени DSA, который выполняет **Request Validation**.

b) **TraceItem.targetObject** устанавливается равным значению **ChainingArguments.targetObject**, если только **object** (или **baseObject** в случае операции Search (поиск)) для аргумента запроса не является идентичным **ChainingArguments.targetObject**, в этом случае **TraceItem.targetObject** должен быть опущен.

c) **TraceItem.operationProgress** устанавливается равным значению **ChainingArguments.operationProgress**.

Если операция была получена от DSA и если **ChainingArguments.streamedResults** содержит значение, большее или равное 1, то следует увеличить значение **ChainingArguments.streamedResults** на 1 тогда и только тогда, когда DSA понимает потоковые результаты и готов принимать потоковые результаты для данной операции.

17.3.4 Утверждение корректности

Операция также должна быть проверена на правильность синтаксиса и семантики согласно правилам, которые содержатся в разделе, где определяется каждая операция (например, необходимо проверить, что **nextRDNToBeResolved** не имеет значение, которое превышает количество RDN в **targetObject**). Если обнаруживается, что запрос содержит неверные аргументы, то операция завершается и пользователю возвращается ошибка, которая зависит от типа обнаруженного несоответствия.

17.3.5 Обнаружение петель

Если в **ChainingArguments.traceInformation** (создается согласно п. 17.3.3) какие-то из двух значений **TraceItem** из являются идентичными, то обработка операции возвратилась к предыдущей стадии и обнаружена петля. В данном случае запрашивающей стороне возвращается **serviceError** (с проблемой **loopDetected**) и **Operation Dispatcher** (диспетчер операций) завершает свою работу.

17.3.6 Неспособность или нежелание выполнять

Процедура **Request Validation** может оценивать имеющиеся ресурсы и затем определить, что операция не может быть выполнена. Кроме этого, также может определяться, на основании соображений стратегии, что операция не должна выполняться. Во всех этих случаях, запрашивающей стороне может возвращаться ошибка **serviceError** (с проблемами **busy**, **unavailable** или **unwillingToPerform**) и диспетчер операций (Operation Dispatcher) завершает свою работу.

Если DSA способен локальными средствами определить, что проблема связана с отсутствие локальных ресурсов DIB, он должен отослать ошибку **serviceError** с проблемой **unavailable**, где компонент **notification** с типом данных **CommonResults** должен содержать следующее:

- атрибут извещения **dSAProblem** со значением **id-pr-dataSourceUnavailable**; и
- атрибут **distinguishedName**, который в качестве значения имеет выделенное имя DSA.

17.3.7 Обработка выходов

В заключительной фазе проверки корректности запроса процедуры **Request Validation** проверенный вход, который был трансформирован с помощью добавления **ChainingArguments** (если был получен от DUA или клиента LDAP) или же путем обновления **ChainingArguments.traceInformation** (если был получен от DSA) возвращается и затем используется как входное значение для процедуры **Name Resolution**.

18 Процедура разрешения имени

18.1 Введение

В данном разделе описывается процедура **Name Resolution**, ее аргументы, результаты и возможные условия возникновения ошибок. Как показывается на рис. 6 (**Operation Dispatcher**), процедура **Name Resolution** состоит из двух процедур:

- процедура **Find DSE**;
- процедура **Name Resolution Continuation Reference**.

Процедура **Find DSE** описывается с помощью трех диаграмм, а именно **Find DSE**, **Target Found** и **Target Not Found**. Процедура **Find DSE** выполняет сопоставление имени целевой статьи и сохраненного локально DSE, она выполняется для всех компонентов. Если цель обнаруживается локально, то **Find DSE** переходит к подпроцедуре **Target Found**, которая затем вызывает процедуру **Check Suitability** с целью проверить, подходит ли найденный DSE для оценки. Если локально обнаружить целевую статью не удается, тогда **Find DSE** переходит к подпроцедуре **Target Not Found** и подготавливает добавление ссылки(ок) продолжения к списку **NRcontinuationList** для того, чтобы он был обработан процедурой **Name Resolution Continuation Reference**.

ПРИМЕЧАНИЕ 1. – При определении сопоставления **Name Resolution** должна выполнить сопоставление имени для множественных выделенных имен, которые различаются в зависимости от контекста, как это описывается в п. 9.4 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

ПРИМЕЧАНИЕ 2. – **Name Resolution** может завершиться неудачей в том случае, если DSA с изданием ниже третьего содержит подчиненную ссылку на статью, которая содержится в DSA более позднего издания и RDN для этой записи содержит контексты. **Name Resolution** завершается неудачей при обработке теневой копии статьи, когда в качестве потенциального имени используется альтернативное имя и теневая статья содержится в DSA, относящегося к первому или второму изданию.

18.2 Параметры процедуры Find DSE

18.2.1 Аргументы

Процедура использует следующие аргументы:

a) **ChainingArguments.traceInformation**;
b) **ChainingArguments.aliasDereferenced**;
c) **ChainingArguments.aliasedRDNs**;
d) **ChainingArguments.excludeShadows**;
e) **ChainingArguments.nameResolveOnMaster**;
f) **ChainingArguments.operationProgress (nameResolutionPhase, nextRDNToBeResolved)**;
g) **ChainingArguments.referenceType**;
h) **ChainingArguments.targetObject**;
i) **ChainingArguments.relatedEntry**;
j) **ChainingArguments.streamedResults**;
k) тип операции;
l) аргумент операции.

ПРИМЕЧАНИЕ. – Если не существует действительных значений, то используются значения по умолчанию или подразумеваемые значения, как это специфицируется в п. 10.3.

18.2.2 Результаты

Существуют два варианта успешного выхода из **Find DSE** (указываются с помощью **entry suitable** или **entry unsuitable**):

Первый успешный выход возвращает (из процедуры **Target Not Found**) ссылку(и) продолжения в списке **NRcontinuationList**, который затем передается процедуре **Name Resolution Continuation Reference** для продолжения фазы разрешения имени.

Второй успешный вариант возвращает (из подпроцедуры **Target Found**) DSE (или ссылку на него), который затем передается одной из процедур оценки.

18.2.3 Ошибки

Могут возвращаться следующие ошибки:

a) **serviceError: unableToProceed, invalidReference, unavailableCriticalExtension, requestedServiceNotAvailable**;

b) **nameError: noSuchObject, aliasDereferencingProblem, contextProblem**.

18.2.4 Глобальные переменные

Процедура использует следующие глобальные переменные:

– список **NRcontinuationList** для хранения ссылки(ок) продолжения, которые необходимы для продолжения разрешения имени в процедуре **Name Resolution Continuation Reference**.

– **StreamedResultsOK**, для того чтобы сохранить возможность определения, может ли данный DSA связывать потоковые результаты в ответ на эту операцию.

18.2.5 Локальные и разделяемые переменные

Процедура использует следующие локальные переменные:

a) **i** индекс, который используется для идентификации компонента целевого имени, которое обрабатывается в данный момент.

b) **m** длина имени целевого объекта, который должен использоваться в разрешении имени. Для тех операций, в которых имя разрешается к статье родителя (например, Add Entry), **m** устанавливается как (количество RDN в целевом объекте – 1). Для всех других операций **m** устанавливается равным количеству RDN в целевом объекте.

c) **lastEntryFound** индекс для DSE(lastEntryFound), который относится к последнему сопоставленному DSE, относящемуся к типу **entry**.

d) **lastCP** индекс для DSE(lastCP), который относится к последнему обнаруженному теневому контекстному префиксу.

e) **candidateRefs** набор ссылок продолжения.

Также используется разделяемая переменная **admPoints** (определена в диспетчере операций). Для удобства компонент **i** имени целевого объекта обозначается как **N(i)**.

18.3 Процедуры

ПРИМЕЧАНИЕ. – В диаграммах присутствует текст, который относится только к определенным операциям. Это не показывается на диаграмме, а сообщается с помощью сопутствующего текстового описания.

18.3.1 Процедура Find DSE

См. рис. 9.

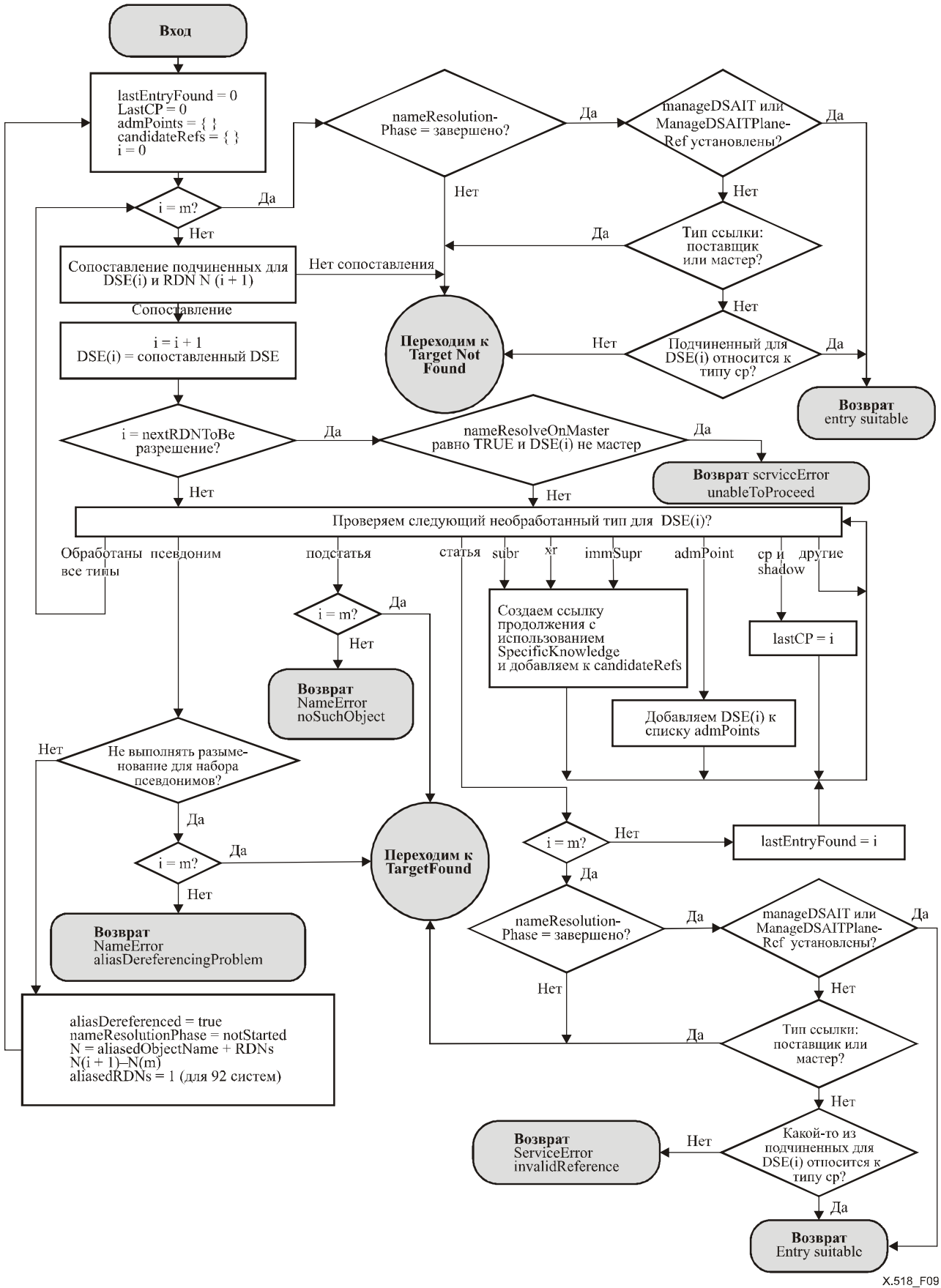


Рисунок 9 – Процедура Find DSE

Имя целевого объекта определяется следующим образом:

a) Если **targetObject** присутствует в **ChainingArguments**, то используется значение этого компонента.

b) Если в **ChainingArguments** присутствует **relatedEntry**, а не **targetObject**, то используется компонент **baseObject** из **JoinArgument**, который идентифицируется с помощью **relatedEntry**.

ПРИМЕЧАНИЕ 1. – Это относится только к защищенному запросу **search**.

c) Если в **ChainingArguments** не присутствуют ни **relatedEntry**, ни **targetObject**, то используется компонент **base** (**baseObject**) данной операции.

Данная процедура пытается локально разрешить имя целевого объекта.

1) Инициализируются локальные переменные **lastEntryFound** и **lastCP** как **0**; **admPoints** и **candidateRefs** как пустой набор, а также инициализируйте **i** как **0**.

2) Сравниваются **i** и **m**. Если они не равны, то переходим к этапу 5).

3) Если же они равны, то проверяется, равно ли **nameResolutionPhase** значению **completed**. Если нет, то переходим к подпроцедуре **Target Not Found**.

Если же **nameResolutionPhase** равна **completed** и установлено критическое расширение **manageDSAIT**, то возвращаемся со значением **entry suitable**.

4) Если **nameResolutionPhase** равно **completed**, то проверяется, является ли какой-то из непосредственные подчиненные DSE(i) контекстным префиксом (типа **cp**).

– Если один (или более) непосредственных подчиненных DSE имеют тип **cp**, то возвращается **entry suitable**.

ПРИМЕЧАНИЕ 2. – Этот случай относится к подзапросам **List (II)** и **Search (II)**.

– Если ни один из непосредственных подчиненных DSE(i) не имеет тип **cp**, то переходим к подпроцедуре **Target Not Found**.

5) Пытаемся найти сопоставление между (**i + 1**)-компонентом из имени целевого объекта с именем подчиненного для последнего сопоставленного DSE. В том случае, если i = 0, то пытаемся сопоставить один из DSE, непосредственно подчиненных корневому DSE. Если сопоставление не обнаруживается, то переходим к подпроцедуре **Target Not Found**. Если обнаруживается одиночное сопоставление, то увеличивается значение **i** и DSE, для которого обнаружено сопоставление, сохраняется как **i**-й элемент в векторе обнаруженных DSE.

ПРИМЕЧАНИЕ 3. – Сопоставление имени включает обработку множественных выделенных имен, которые различаются в зависимости от контекста (там, где он известен) как описывается в п. 9.4 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

Если обнаруживается более чем одно сопоставлении, то возвращается **nameError** с проблемой **contextProblem**.

ПРИМЕЧАНИЕ 4. – Например, это может происходить в том случае, когда **AttributeTypeAndDistinguishedValue** в потенциальном имени содержит значения множественных выделенных имен, который различаются в зависимости от контекста и некоторые из этих значений сопоставляются в различных целевых именах.

6) Если **i** равняется **nextRDNToBeResolved**, то проверяем, выполняются ли следующие два условия:

– аргумент **ChainingArgument.nameResolveOnMaster** равен **TRUE**;

– DSE(**i**) не является главной статьей.

Если оба этих условия удовлетворяются, то возвращается **serviceError** с проблемой **unableToProceed**.

ПРИМЕЧАНИЕ 5. – Это означает, что используется **nameResolveOnMaster** с целью избежать нескольких путей к тому же целевому объекту.

7) Проверить все биты типа DSE в DSE(**i**). Для каждого бита типа может понадобиться провести определенную обработку. Ниже описывается, какие действия необходимо предпринять для каждого обнаруженного типа:

– Если установлены биты **cp** и **shadow**, то индекс **i** сохраняется в **lastCP**.

– Если установлен бит **admPoint**, то проверьте операционный атрибут **administrativeRole**. Если это начало автономной административной области, то необходимо очистить список **admPoints**. Если это начало одной или нескольких специальных административных областей, тогда проверьте список **admPoints** и удалите все существующие точки, которые уже не представляют интереса (например, их роли были заменены новыми административными точками). Сохраняем DSE (**i**) в списке.

– Если установлен один из битов **subr**, **xr**, **immSupr** или **ditBridge**, то создается ссылка продолжения с использованием атрибута **specificKnowledge** со значением **operationProgress.nameResolutionPhase**, установленным как **proceeding**, **nextRDNToBeResolved** равным **i**, **targetObject** созданным из разрешенных компонентов с использованием первичных RDN (в RDN могут включаться альтернативные выделенные имена), которые объединяются с остающимися неразрешенными компонентами, при этом **accessPoints** и **referenceType** устанавливаются нужным образом. Добавляем ссылку продолжения в список ссылок продолжения в **candidateRefs**.

– Если установлен бит **entry**, то проверяем на равенство **i** и **m** (и, следовательно, имя целевого объекта является полностью сопоставленным). Если **i** не равняется **m**, то найденная запись запоминается с помощью установки значения **lastEntryFound** равным **i** и затем продолжается обработка битов типа в DSE(**i**). Если **i** равно **m**, то переходим к этапу 8).

– Если установлен бит **subentry**, то проверяем, что **i** равно **m** (и, следовательно, имя целевого объекта является полностью сопоставленным). Если они равны, то переходим к процедуре **Target Found**; если они не равны, то возвращается **nameError** с проблемой **noSuchObject**.
– Если бит **alias** установлен, то проверяем, установлен ли бит **dontDereferenceAliases**.

  Если **dontDereferenceAliases** не установлен, то псевдоним может быть разыменован. Следовательно, установите **chainingArguments.aliasDereferenced** равным **TRUE**, **nameResolutionPhase** равным **notStarted**, название целевого объекта в качестве значения **aliasedEntryName**, как оно передается в статье псевдонима, в сочетании с остающимися несопоставленными компонентами предыдущего имени целевого объекта (т. е. объединяем с компонентами от (**i + 1**) до **m** предыдущего имени целевого объекта). Начиная с второго издания DSA не устанавливают **aliasedRDNs** (в то время как DSA первого издания устанавливают **aliasedRDNs** равным количеству RDN в **aliasedEntryName**). Вновь запустите разрешение имени с помощью перехода к этапу 1).

  Если установлен **dontDereferenceAliases**, тогда псевдоним не может быть разыменован. Проверьте, что имя целевого объекта было обработано полностью, проверяя на равенство значения **i** и **m**. Если они равны (и таким образом полностью сопоставлены), то переходим к подпроцедуре **Target Found**. Если они не равны, (и таким образом имена сопоставлены не полностью), то возвращается **nameError** с проблемой **aliasDereferencingProblem**.
– Для всех других возможных типов DSE не требуется никакого действия. Внутренним образом данный тип DSE отмечается как обработанный и продолжается обработка необработанных на данный момент битов типа DSE для DSE(**i**).
– Если обработаны все биты типа для DSE(**i**), то переходим к стадии 2).

8) Проверяем, что **nameResolutionPhase** равна **completed**. Если это неверно, то переходим к подпроцедуре **Target Found**.
9) Если **nameResolutionPhase** уже равно **completed** и установлено значение для критического расширения **manageDSAIT**, то возвращаемся с **entry suitable**.
10) В противном случае проверяем, является ли какой-то из DSE, которые непосредственно подчиняются DSE(**i**), контекстным префиксом (и, следовательно, имеет тип **cp**). Если такие имеются (один или более), то возвращаем **entry suitable**. Если же ни одна из непосредственно подчиненных статей не относится к типу контекстного префикса, то возвращается **serviceError** с проблемой **invalidReference**.

ПРИМЕЧАНИЕ 6. – Этот случай относится к подзапросам **List (II)** и **Search (II)**.

18.3.2 Подпроцедура Target Not Found

См. рис. 10.

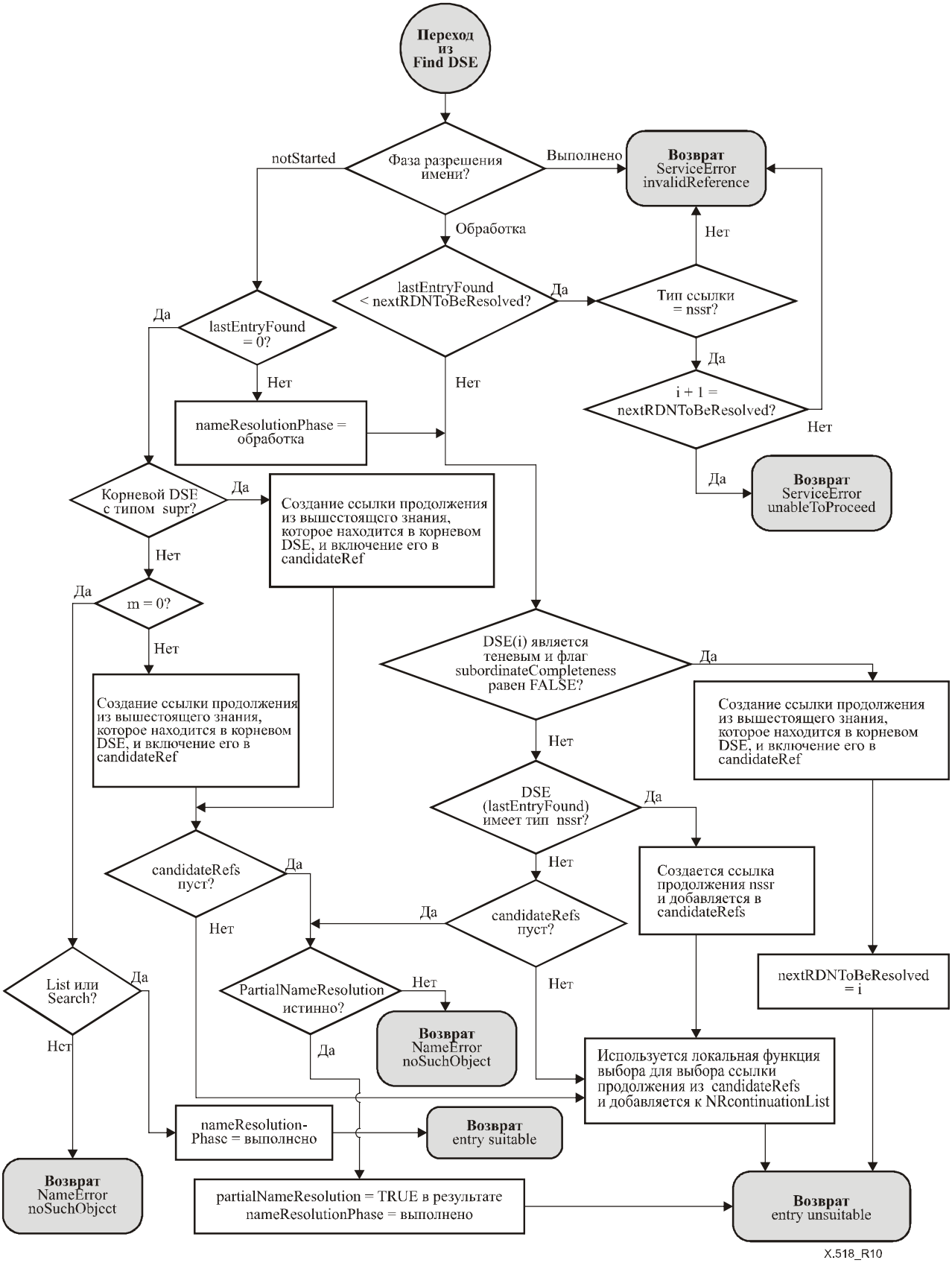


Рисунок 10 – Подпроцедура Target Not Found

Данная подпроцедура вызывается в том случае, если в локальном DSA не удается найти имя целевого объекта. Подпроцедура определяет, какой тип ссылки на знание лучше использовать для продолжения разрешения имени, если только не была обнаружена ошибка – в этом случае возвращается ошибка.

1) При продолжении после окончания процедуры **Find DSE**, необходимо различать между тремя различными состояниями для фазы разрешения имени.
   - если **nameResolutionPhase** равно **notStarted**, то переходим к этапу 2).
   - если **nameResolutionPhase** равно **proceeding**, то переходим к этапу 8).
   - если **nameResolutionPhase** равно **completed**, то переходим к этапу 12).
2) Если статья найдена (**lastEntryFound** не равно **0**), устанавливаем значение **nameResolutionPhase** равным **proceeding** и переходим к этапу 9).
3) Если статья не найдена (**lastEntryFound=0**), то проверьте, относится ли данный DSA к DSA первого уровня.

   Если это DSA первого уровня, то корневой DSE не содержит вышестоящую ссылку и таким образом не относится к типу **supr**. В этом случае переходим к этапу 4).

   Если же DSA не является DSA первого уровня, то корневой DSE содержит старшую ссылку и, таким образом, относится к типу **supr**. В этом случае создаем ссылку продолжения с использованием старшего знания, которое хранится корневым DSE. Устанавливаем:
   - значение **targetObject** устанавливаем равным имени целевого объекта, который конструируется из разрешенных компонентов с использованием первичных RDN (в RDN могут включаться альтернативные выделенные значения), с которыми объединяются остающиеся неразрешенные компоненты;
   - значение **operationProgress.nameResolutionPhase** устанавливаем равным **notStarted**;
   - значение **referenceType** устанавливаем равным **superior**; и
   - значение **accessPoints** устанавливаем соответствующим образом.

   Добавляем ссылку продолжения к списку ссылок продолжения в **candidateRefs**. Переходим к этапу 6).
4) Проверяем, не предназначена ли данная операция для корневой записи (**m = 0?**). Если это так, то переходим к этапу 5). В противном случае, создаем ссылку продолжения с использованием любого знания NSSR, который находится в корневом DSE. Устанавливаем:
   - значение **targetObject** устанавливаем равным целевому объекту, который конструируется из разрешенных компонентов с использованием первичных RDN (в RDN могут включаться альтернативные выделенные имена), с которыми соединяются остающиеся неразрешенные компоненты;
   - значение **operationProgress.nameResolutionPhase** равным **proceeding**;
   - значение **operationProgress.nextRDNToBeResolved** равным **1**;
   - значение **referenceType** равным **nonSpecificSubordinate**; и
   - значение **accessPoints** устанавливаем соответствующим образом.

   Добавляем ссылку продолжения к списку ссылок продолжения в **candidateRefs**. Переходим к этапу 6).
5) Для DSA первого уровня в тех случаях, когда в качестве базового объекта используется корневая запись, могут выполняться только операции List (список) или Search (поиск). Таким образом, если операция не является операцией List (список) или Search (поиск), то возвращаем **nameError** с проблемой **noSuchObject**. Если же эта операция является операцией List (список) или Search (поиск), то устанавливаем **nameResolutionPhase** равным **completed** и возвращаем **entry suitable**.
6) Проверяем, имеются ли в **candidateRefs** ссылки продолжения. Если **candidateRefs** пуст и **partialNameResolution** равно **FALSE**, то возвращается **nameError** с проблемой **noSuchObject**. Если **candidateRefs** пуст и **partialNameResolution** равен **TRUE**, то в результате устанавливаем значение **partialName** равным **TRUE**, **nameResolutionPhase** равным **completed**, и возвращаем **entry suitable**. В противном случае переходим к этапу 7).
7) Используем функцию локального выбора для выбора ссылки продолжения из списка ссылок продолжения в **candidateRefs**, добавляем ее к списку ссылок продолжения в **NRcontinuationList** и возвращаем вместе с **entry unsuitable**.
8) Если DSA не сумел обработать разрешение имени (в этом случае **lastEntryFound** имеет значение меньшее, чем **nextRDNToBeResolved**), переходим к этапу 11). В противном случае переходим к следующему этапу.
9) Если DSE(**i**) является теневым DSE с неполным знанием о подчиненном (**subordinateCompletenessFlag** равен **FALSE**), то создается ссылка продолжения на основании атрибута **supplierKnowledge**, который содержится в **DSE(lastCP)**. Устанавливаем:
   - значение **targetObject** устанавливается равным целевому объекту, который конструируется на основании первичных RDN (в RDN могут включаться альтернативные выделенные имена) из компонентов, для которых выполнено разрешение, и которые объединяются с остающимися неразрешенными компонентами;

– значение **operationProgress.nameResolutionPhase** устанавливается равным **proceeding**;
– значение **operationProgress.nextRDNToBeResolved** устанавливается равным **lastEntryFound**;
– значение **referenceType** устанавливается равным **supplier**; и
– значение **accessPoints** устанавливается соответствующим образом.

Ссылка продолжения добавляется к списку ссылок продолжений в **NRcontinuationList** и возвращаемся с **entry unsuitable**.

10) Если последняя обнаруженная запись содержит NSSR (**DSE(lastEntryFound)** относится к типу **nssr**), то создается ссылка продолжения из знания NSSR, которое хранится в **DSE(lastEntryFound)**. Устанавливаем:
– значение **targetObject** устанавливается равным целевому объекту, который конструируется на основании первичных RDN (в RDN могут включаться альтернативные выделенные имена) из компонентов, для которых выполнено разрешение, и которые объединяются с остающимися неразрешенными компонентами;
– значение **operationProgress.nameResolutionPhase** устанавливается равным **proceeding**;
– значение **operationProgress.nextRDNToBeResolved** устанавливается равным **lastEntryFound+1**;
– значение **referenceType** устанавливается равным **nonSpecificSubordinate**; и
– значение **accessPoints** устанавливается соответствующим образом.

Ссылка продолжения добавляется к списку ссылок продолжений в **candidateRefs**. Затем переходим к этапу 7).

Если **DSE(lastEntryFound)** не относится к типу **nssr**, то переходим к этапу 6).

11) Если **chainingArguments.referenceType** относится к типу **nssr**, тогда переходим к этапу 13), в противном случае переходим к этапу 12).
12) Возвращаем значение **serviceError** с проблемой **invalidReference**.
13) Если **i + 1** равняется **nextRDNToBeResolved**, то запрос был направлен NSSR и DSA не способен обработать разрешение имени; в этом случае возвращаем **serviceError** с проблемой **unableToProceed**; в противном случае переходим к этапу 12).

18.3.3 Подпроцедура Target Found

Данная подпроцедура используется тогда, когда имя целевого объекта локально совпадает со статьей DSE. Подпроцедура проверяет, подходит ли найденная запись для локальной обработки запроса (которая показана на рис. 11):

1) Вызывается процедура Check Suitability (проверка соответствия).
2) Если запись соответствует (**entry suitable**), то выполняем следующее:
– устанавливаем **nameResolutionPhase** равным **completed**;
– сравниваем значение **ChainingArguments.streamedResults** (если существует) с количеством элементов, содержащихся в **ChainingArguments.traceInformation**; если они равны, то устанавливаем StreamedResultsOK равным true; и
– возвращаем **entry suitable**.
3) Если запись не соответствует (**entry unsuitable**), то создаем ссылку продолжения на основании атрибута **supplierKnowledge**, который содержится в **DSE(lastCP)**. Устанавливаем:
– значение **targetObject** устанавливается равным целевому объекту, который конструируется на основании первичных RDN (в RDN могут включаться альтернативные выделенные имена) из компонентов, для которых выполнено разрешение, и которые объединяются с остающимися неразрешенными компонентами;
– значение **operationProgress.nameResolutionPhase** устанавливается равным **proceeding**;
– значение **operationProgress.nextRDNToBeResolved** устанавливается равным **m**;
– значение **referenceType** устанавливается равным **supplier**; и
– значение **accessPoints** устанавливается соответствующим образом.

Добавляем ссылку продолжения к списку ссылок продолжений в **NRcontinuationList**. Возвращаем **entry unsuitable**.

ПРИМЕЧАНИЕ. – Если установлено управление службой **localScope**, то DSA может, на основании локальных стратегий, принять решение о рассмотрении данной статьи как соответствующей и перейти к обработке как в этапе 2).

4) Если критическое расширение не поддерживается (**unsupported critical extension**), то возвращаем **serviceError** с проблемой **unavailableCriticalExtension**.

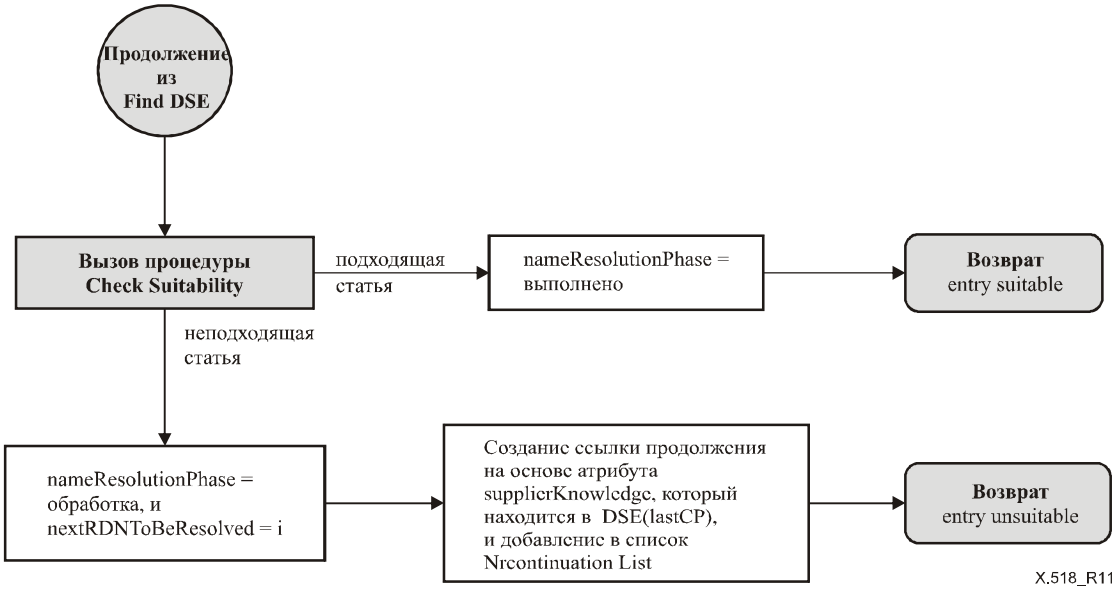


Рисунок 11 – Процедура Target Found

18.3.4 Процедура Check Suitability (проверка соответствия)

Данная процедура вызывается тогда, когда необходимо определить, является ли обнаруженный DSE соответствующим для выполнения запрашиваемой операции (см. рис. 12). При этом учитываются значения **ChainingArguments**, **ServiceControls**, предоставленные пользователем аргументы, тип операции и характеристики DSE (тень, подчиненное знание, имеющиеся атрибуты и так далее).

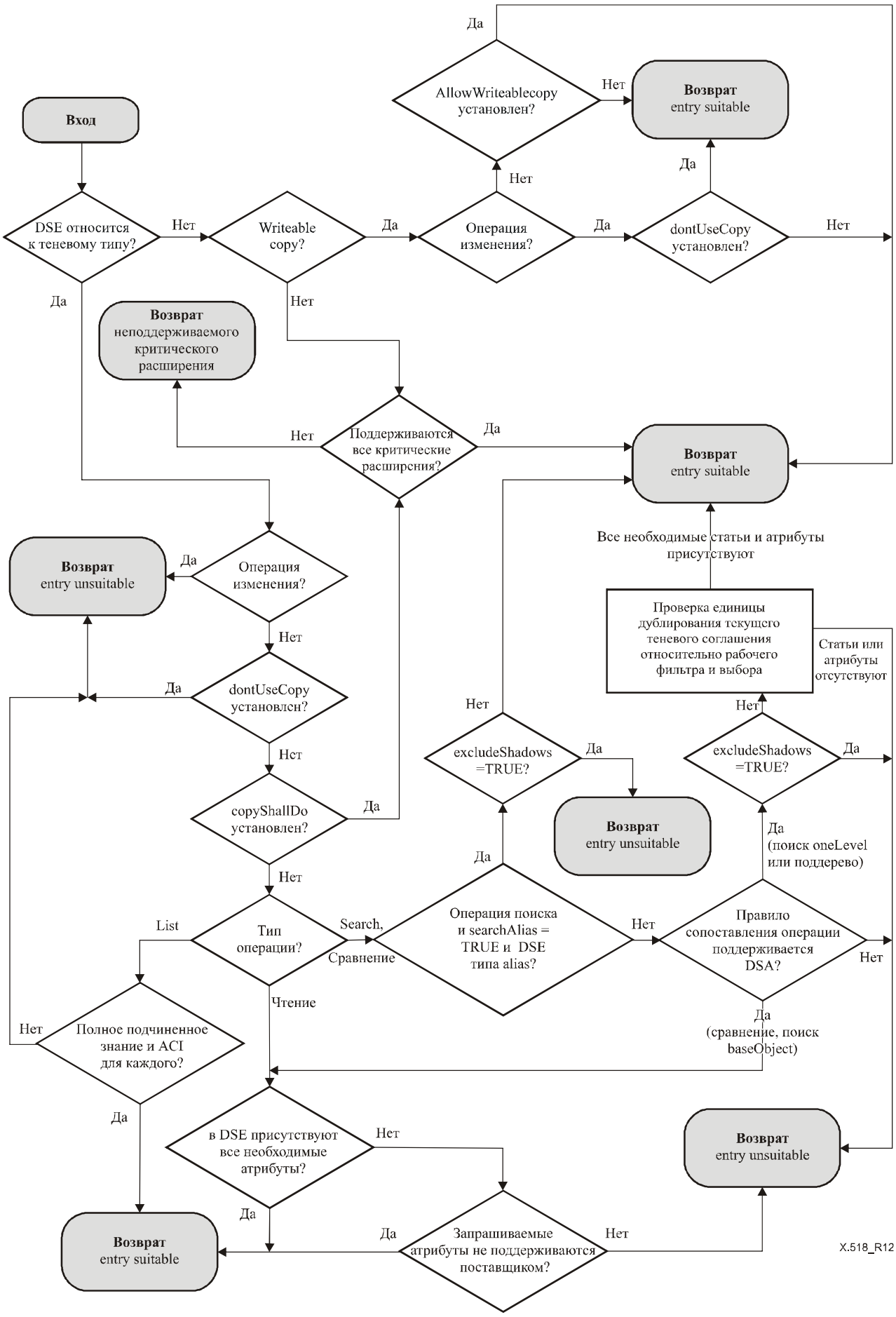


Рисунок 12 – Процедура Check Suitability

18.3.4.1 Параметры процедуры

В качестве входных аргументов для этой процедуры используются:

- ссылка на DSE;
- тип операции, для которой необходимо проверить соответствие DSE;
- аргументы связывания **ChainingArguments**; и
- аргумент операции.

Выходными значениями могут являться **entry suitable**, **entry unsuitable** или **unsupported critical extension**.

1) Если DSE не относится к типу **shadow** и не относится к типу **writeableCopy**, то проверяем поддержку всех **criticalExtensions**. Если это выполняется, то возвращаем, что запись соответствует, в противном случае возвращается неподдерживаемое критическое расширение.
2) DSE относится к типу **shadow**. Возвращаем, что запись не соответствует, если выполняется одно из следующих условий:
   - тип запрошенной операции относится к операциям изменения.
   - установлено управление службой **dontUseCopy**.

   В противном случае переходим к следующему этапу.
3) Если DSE относится к типу **writeableCopy**, возвращается информация о том, что статья не соответствует, если выполняется одно из следующих условий:
   - тип запрошенной операции относится к операциям изменения.
   - тип запрошенной операции относится к операции запроса и не установлено управление службой **allowWriteableCopy**.

   В противном случае возвращается информация о том, что статья соответствует.
4) Если установлено управление службой **copyShallDo**, то проверяем, поддерживаются ли все расширения **criticalExtensions**. Если они поддерживаются, то возвращается **entry suitable**, в противном случае возвращается **unsupported critical extension**.
5) Если не установлено управление службой **copyShallDo**, то проверяем, поддерживаются ли все расширения **criticalExtensions**. Если они поддерживаются, то переходим к этапу 5), в противном случае возвращаем **entry unsuitable**.
6) В зависимости от типа операции:

   Для операции List (список) переходим к этапу 6).

   Для операции Read (читать) переходим к этапу 7).

   Для операций Search (поиск) или Compare (сравнить) переходим к этапу 8).
7) Если статья имеет полное подчиненное знание, то может быть выполнена операция List (список). В этом случае возвращаем **entry suitable**, в противном случае возвращаем **entry unsuitable**.
8) Если все запрашиваемые атрибуты присутствуют в DSE, то возвращаем **entry suitable**. Если некоторые из атрибутов отсутствуют, то локальными средствами определяем, содержит ли теневая копия все те атрибуты, которые содержит мастер (например, с помощью ссылки на теневое соглашение). Если содержит, то запись соответствует (возвращается **entry suitable**). В противном случае, поставщик может хранить те запрашиваемые атрибуты, которые не содержатся в теневом объекте (shadow); в этом случае для запроса необходимо связывание (возвращаем **entry unsuitable**).
9) Если операция является операцией **search**, значение **searchAliases** установлено как **TRUE** и DSE относится к типу **alias**, тогда если значение **chainingArguments.excludeShadows** равно **FALSE**, то возвращаем **entry suitable**, если же равно **TRUE**, то возвращаем **entry unsuitable**.
10) Если DSA поддерживает правило сопоставления для поиска или сравнения в соответствии с запросом, и операцией является **compare** или **search** с **subset** для **baseObject**, то переходим к этапу 7). Если же DSA поддерживает правило сопоставления и операцией является **search** с подмножеством **oneLevel** или **subtree**, то переходим к этапу 10). В противном случае возвращаем **entry unsuitable**.
11) Если **chainingArguments.excludeShadows** равен **TRUE**, то возвращаем **entry unsuitable**. В противном случае проверяем, понятна ли локально спецификация теневой информации для фильтра и выбора операции. Если все необходимые статьи и атрибуты присутствуют, то возвращаем **entry suitable**. Если статья или атрибут отсутствуют, то возвращаем **entry unsuitable**.

19 Оценка операции

В данном разделе определяются процедуры, которым должен следовать DSA в том случае, если для данной операции целевая статья обнаруживается локально (во время разрешения имени). В зависимости от типа операции может вызываться одна из следующих процедур:

- Для операций **addEntry**, **chainedAddEntry**, **removeEntry**, **chainedRemoveEntry**, **modifyEntry**, **chainedModifyEntry**, **modifyDN** или **chainedModifyDN** необходимо следовать процедурам из п. 19.1.
- Для операций **read**, **chainedRead**, **compare** или **chainedCompare** необходимо следовать процедурам из п. 19.2.
- Для операций **search**, **chainedSearch**, **list** или **chainedList** необходимо следовать процедурам из п. 19.3.

19.1 Процедура изменения

В соответствии с типом операции изменения необходимо следовать процедурам, определенным в пп. 19.1.1–19.1.4.

19.1.1 Операция Add Entry (добавление статьи)

1) DSA должен проверить, что инициатор обладает соответствующими правами доступа, например, согласно п. 11.1.5 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если нет, то возвращается соответствующая ошибка.
2) DSA должен удостовериться, что не существует статьи с таким же именем как у статьи, которая должна быть добавлена. В противном случае он должен возвратить **updateError** с проблемой **entryAlreadyExists**. Если вышестоящий DSE относится к дополнительному типу **nssr**, то DSA должен действовать в соответствии с процедурой, определенной в п. 19.1.5 (операции Modify (изменить) и NSSR) для того, чтобы удостовериться, что имя новой статьи является однозначным. Если имя статьи, которая должна быть добавлена, включает множественные выделенные значения, которые различаются в зависимости от содержания для какого-то атрибута в конечном RDN, то DSA должен удостовериться, что ни одно из возможных альтернативных имен RDN, которые могут быть сконструированы, не выдает (вне зависимости от содержания) имя уже существующей статьи.
3) Если присутствует **targetSystem** и **AccessPoint** не относится к текущему DSA, то переходим к этапу 4). Если **targetSystem** отсутствует или же присутствует, но **AccessPoint** относится к текущему DSA, то переходим к этапу 5).
4) Если статья является подстатьей, то DSA возвращает **updateError** с проблемой **affectsMultipleDSAs**. Если статья не является подстатьей, то DSA имеет возможность локального выбора: следует или нет устанавливать иерархическое операционное связывание HOB со специфицированным DSA. Если не следует, то DSA возвращает **serviceError** с проблемой **unwillingToPerform**, в противном случае DSA устанавливает иерархическое операционное связывание (HOB, hierarchical operational binding) со специфицированным подчиненным DSA. Если DOP не поддерживается, то действуем согласно процедуре из п. 24.3.1.1. В противном случае, для установления HOB используются локальные средства. Если подчиненный DSA не хочет устанавливать операционное связывание, то для операции **addEntry** возвращается **serviceError** с проблемой **unwillingToPerform**. Если HOB устанавливается успешно, то переходим к этапу 7).

   ПРИМЕЧАНИЕ 1. – Данный этап процедуры не применяется при создании автономных административных областей в подчиненном DSA.
5) DSA должен удостовериться, что новая статья соответствует подсхеме, или же что новая подстатья или DSE другого типа соответствуют схеме системы (например, что непосредственный вышестоящий DSE для подстатьи относится к типу **admPoint**). Если это не верно, то он должен возвратить соответствующую ошибку **updateError** или **attributeError**, в противном случае он добавляет новый DSE. Если добавляется статья, то переходим к этапу 7). Если добавляется подстатья, то переходим к этапу 6). В противном случае для других типов DSE выполняются соответствующие процедуры управления знаниями. См. раздел 6.
6) В соответствующее время DSA должен направить операционное связывание изменения всем релевантным подчиненным DSA, с которыми он имеет иерархические или неспецифические иерархические операционные связывания. Релевантное связывания – это те, которые ассоциируются с контекстами именования, которые являются подчиненными для вышестоящего DSE. Контексты именования, у которых контекстные префиксы соответствуют автономным административным точкам, не являются релевантными. Если поддерживается DOP, то следует действовать в соответствии с процедурами из пп. 24.3.2.1 и 25.3.2. Если DOP не поддерживается, то для модификации RHOB следует использовать локальные средства.

   ПРИМЕЧАНИЕ 2. – Администратор DSA определяет время, которое является соответствующим, его диапазон может простираться от непосредственно после (или даже до) того времени, когда возвращается результат для периодической стратегии (например, в назначенный час). Время может изменяться в зависимости от причины, которая вызвала изменение, например, обновления в ACI оказывают мгновенный эффект, а изменения в схеме выполняются периодически.
7) Если добавленная статья или подстатья находится внутри **UnitOfReplication** для одного или нескольких теневых соглашений, тогда теневые потребители будут обновляться с использованием процедур информационной теневой службы Справочника, которые определены в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

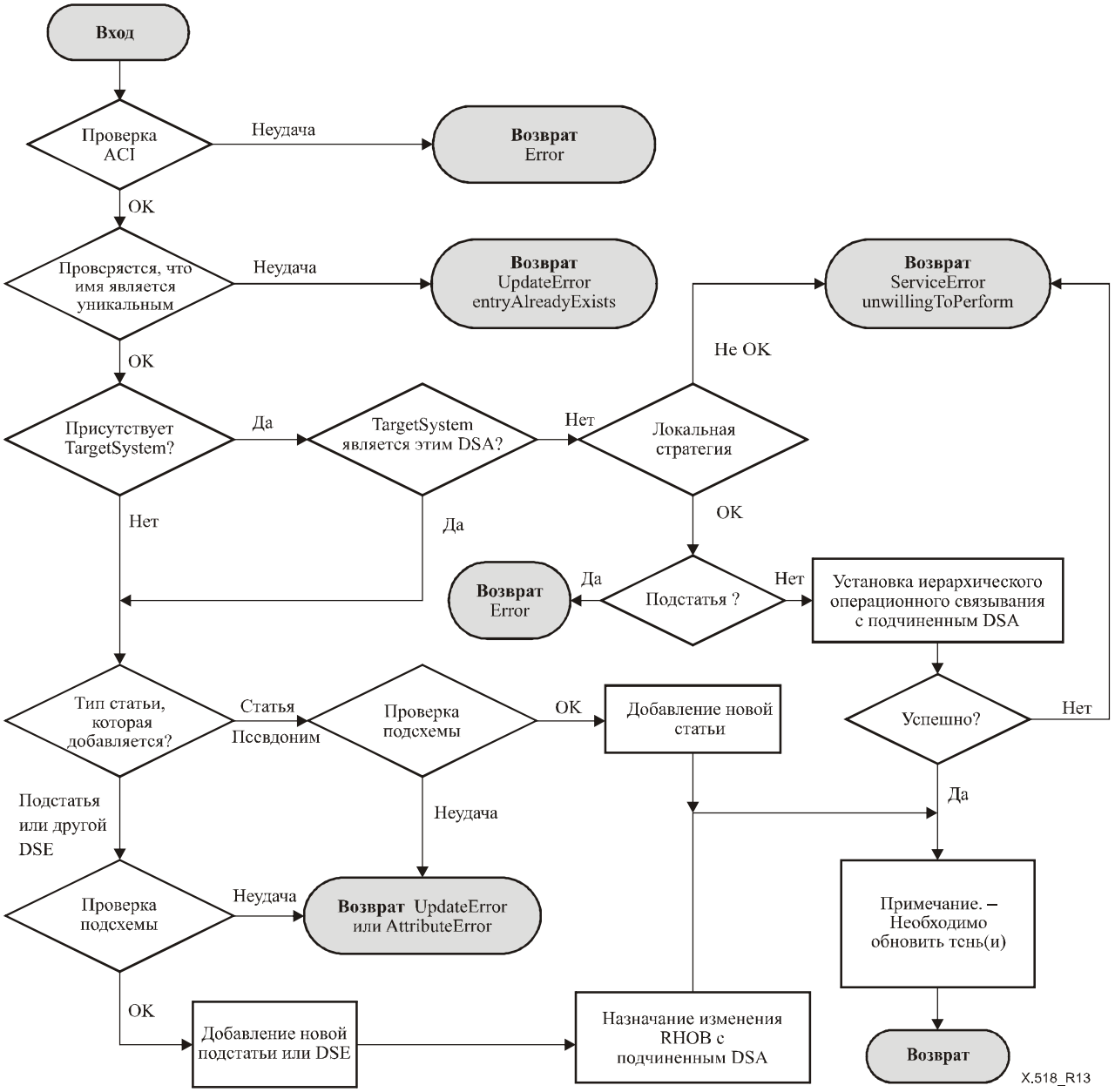


Рисунок 13 – Процедура Add Entry

19.1.2 Операция Remove Entry (удаление статьи)

1) DSA должен проверить, что инициатор обладает достаточными правами доступа, например, в соответствии с определениями п. 11.2.5 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если нет, то возвращается соответствующая ошибка.
2) DSA должен удостовериться в том, что статья, которая должна быть удалена, является листовой статьей (leaf entry). В противном случае, DSA возвращает **updateError** с проблемой **notAllowedOnNonLeaf**.
3) Проверяется тип DSE для статьи, которая должна быть удалена. Если тип соответствует **subentry**, то переходим к этапу 5). Если тип соответствует **cp**, то переходим к этапу 6). Если тип соответствует **entry** или **alias**, то переходим к этапу 4). В противном случае, для других типов DSE выполняется соответствующая процедура управления знанием. См. раздел 6.
4) Удаляем статью или статью псевдонима и переходим к этапу 7).
5) Удаляем подстатью. В соответствующее время изменяем операционные связывания для всех релевантных подчиненных DSA, с которыми текущий DSA имеет иерархические или неспецифические иерархические связывания. Релевантные связывания – это те, которые ассоциируются с контекстами именования, которые являются подчиненными для вышестоящего DSE. Контексты именования, контекстные префиксы которых соответствуют автономным административным точкам, не являются релевантными. Если поддерживается DOP, то следует действовать в соответствии с процедурами из пп. 24.3.2.1 и 25.3.2. Если DOP не поддерживается, то следует использовать локальные средства. Переходим к этапу 7).

6) Удаляем контекст именования. Если DSA имеет иерархическое операционное связывание для данного контекста именования, то он должен завершить иерархическое операционное связывание со своим непосредственно вышестоящим DSA. Если DSA имеет неспецифическое иерархическое операционное связывание для данного контекста именования и он является последним контекстом именования для данного неспецифического иерархического связывания, то он должен завершить неспецифическое иерархическое операционное связывание со своим непосредственно вышестоящим DSA. Если поддерживается DOP, то следует действовать в соответствии с процедурами из пп. 24.3.3.2 и 25.3.3.2. В противном случае для завершения RHOB должны использоваться локальные средства.

7) Если удаленный контекст именования, статья, статья псевдонима или подстатья относились к **UnitOfReplication** для одного или более теневых соглашений, тогда теневые потребители будут обновляться с использованием процедур информационной теневой службы Справочника, которые определены в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

   Если удаленная подчиненная или неспецифическая подчиненная ссылка в непосредственно вышестоящем DSA (RHOB которого был завершен) находится в **UnitOfReplication** для одного или нескольких теневых соглашений, тогда теневые потребители будут обновляться с использованием процедур информационной теневой службы Справочника, которые определены в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

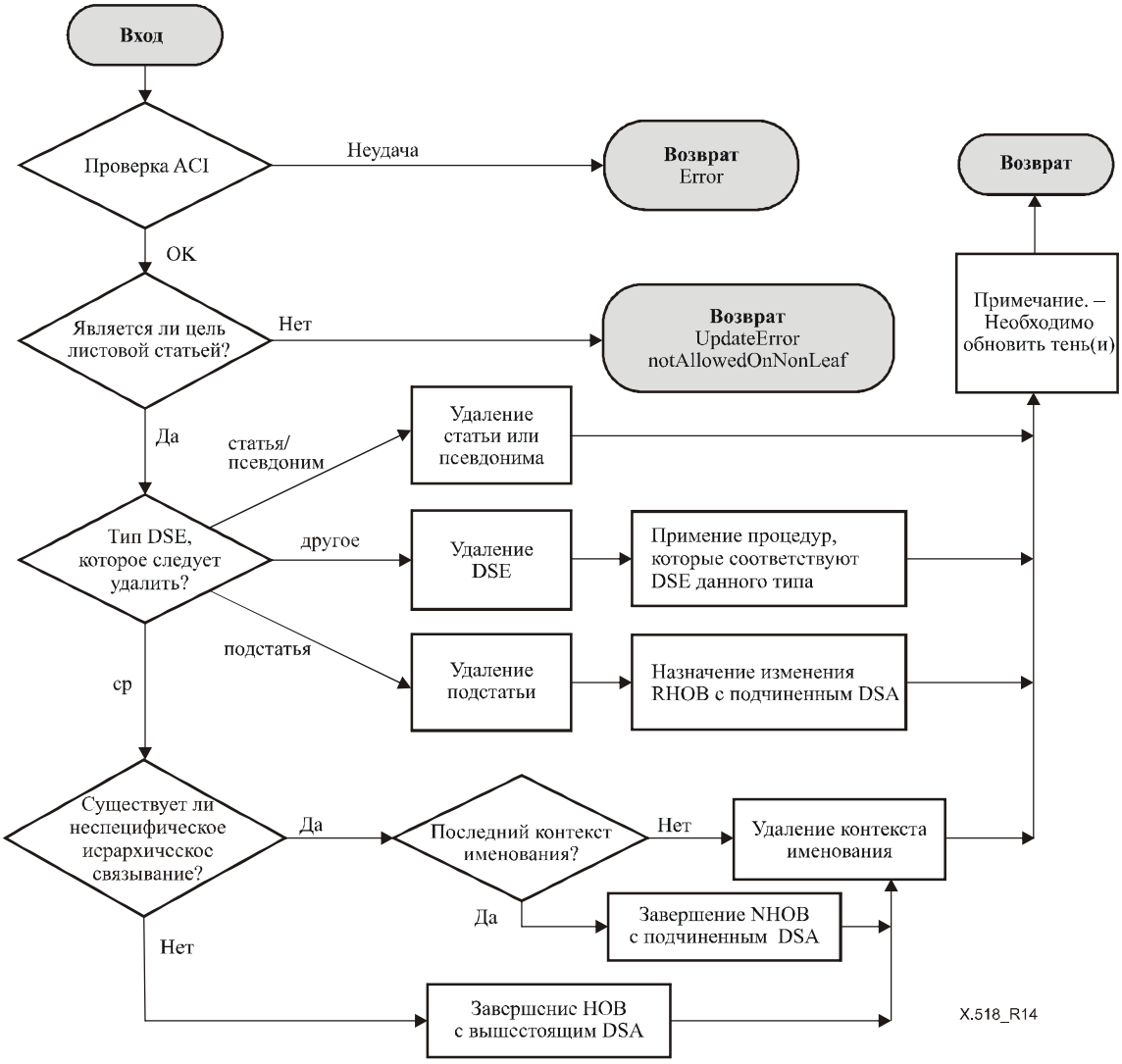


Рисунок 14 – Процедура Remove Entry

19.1.3 Операция Modify Entry (изменение статьи)

1) DSA должен проверить, что инициатор обладает достаточными правами доступа, например, как это определено в п. 11.3.5 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если не обладает, то возвращается соответствующая ошибка.

2) Изменение статьи или псевдонима должно соответствовать подсхеме. Изменение DSE других типов, включая подстатью, должно соответствовать системной схеме. В противном случае DSA возвращает соответствующую ошибку **updateError** или **attributeError**. Если целевая DSE относится к типу **subentry**, то после выполнения модификаций переходим к этапу 3); если целевая DSE относится к типу **entry** или **alias**, то переходим к этапу 4); в противном случае, для других типов DSE выполняются соответствующие процедуры управления знанием. См. раздел 6.

3) В соответствующее время DSA должен изменить операционное связывание со всеми релевантными подчиненными DSA, с которыми он имеет иерархические или неспецифические иерархические операционные связывания. Релевантные связывания – это те, которые ассоциируются с контекстами именования, которые являются подчиненными для административной точки, ниже которой располагается модифицируемая статья. Контексты именования, контекстные префиксы которых соответствуют автономным административным точкам, не являются релевантными. Если поддерживается DOP, то следует действовать в соответствии с процедурами из пп. 24.3.2.1 и 25.3.2. Если DOP не поддерживается, то для модификации RHOB следует использовать локальные средства.
4) Если измененная статья, статья псевдонима или подстатья находилась внутри **UnitOfReplication** для одного или более теневых соглашений, то теневые потребители должны быть обновлены в соответствии с процедурами информационной теневой службы Справочника, которые специфицируются в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

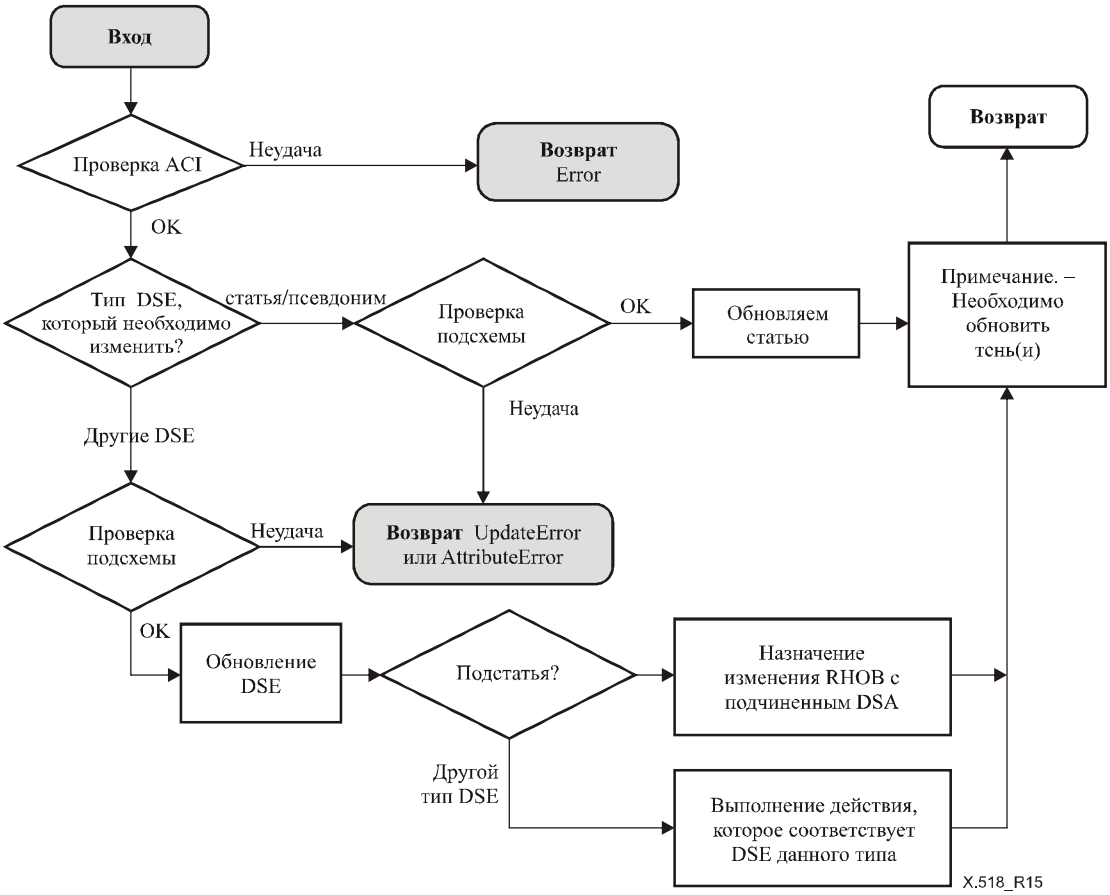


Рисунок 15 – Процедура Modify Entry

19.1.4 Операция Modify DN (изменение DN)

1) DSA должен проверить, что инициатор обладает достаточными правами доступа, например, как это определено в п. 11.3.5 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если не обладает, то возвращается соответствующая ошибка.
2) Если операцией является перемещение статьи или же совместное перемещение статьи и изменение ее относительного выделенного имени (Relative Distinguished Name), то переходим к этапу 3). Если же операция заключается в изменении относительного выделенного имени для статьи, то переходим к этапу 4).
3) Операция должна выполняться в соответствии с определением п. 11.4.1 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если прежний вышестоящий, новый вышестоящий, статья или любой из их подчиненных не относится к данному DSA, или же если новый вышестоящий имеет NSSR, то операция отвергается с ошибкой **updateError** и проблемой **affectsMultipleDSAs**. DSA должен удостовериться, что не существует статей с именем, соответствующим новому имени. В противном случае возвращается ошибка **updateError** с проблемой **entryAlreadyExists**. DSA должен удостовериться, что новое имя статьи соответствует подсхеме. В противном случае возвращается соответствующая ошибка **attributeError** или **updateError**. Если ни одна из этих проблем не возникает, то перемещаем статью (при необходимости также изменяем RDN) и переходим к этапу 9).
4) Следующий далее текст относится к изменению относительного выделенного имени для статьи, которая может либо являться, либо не являться листовой записью и которая может иметь или же не иметь одного или более подчиненных в одном или более DSA. Проверяется тип DSE для записи, которую следует переименовать. Если тип соответствует **subentry**, то переходим к этапу 7). Если **cp**, то переходим к этапу 6). Если тип соответствует **entry** или **alias**, переходим к этапу 5).

5) DSA должен удостовериться, что не существует статей с именем, которое совпадает с новым именем. В противном случае возвращается ошибка **updateError** с проблемой **entryAlreadyExists**. Если для статьи, подлежащей переименованию, DSE относится к дополнительному типу **nssr**, то DSA должен следовать процедуре, определенной в п. 19.1.5 (Операции изменения и NSSR) с той целью, чтобы удостовериться, что имя данной записи является уникальным. Если новое имя включает множественные выделенные имена, которые различаются в зависимости от контекста для некоторого атрибута в RDN, то DSA должен удостовериться, что ни один из возможных RDN, которые могут быть сконструированы (независимо от содержания), может получать название, которое совпадает с названием уже существующей статьи. DSA должен гарантировать, что новое имя статьи соответствует подсхеме. В противном случае он должен возвратить соответствующую ошибку **attributeError** или **updateError**. Изменяем название статьи или статьи псевдонима. Если статья не относится к листовым статьям и имеет подчиненных в других DSA, переходим к этапу 8), в противном случае переходим к этапу 9).

6) DSA должен гарантировать, что новое название контекста именования соответствует подсхеме; в противном случае он должен возвратить соответствующие ошибки **attributeError** или **updateError**.

   Если DSA обладает HOB с вышестоящим DSA, тогда подчиненный DSA должен попытаться модифицировать HOB перед тем, как реагировать на операцию изменения DN (Modify DN). Перед тем, как принять изменения, вышестоящий DSA должен гарантировать, что не существует статьи с аналогичным именем. Если поддерживается DOP, то необходимо следовать процедуре из п. 24.3.2.2. Если же DOP не поддерживается, то модификация HOB является локальной задачей и новое имя проверяется на уникальность. Если модификация HOB происходит успешно, и контекст именования имеет подчиненные контексты именования в других DSA, то переходим к этапу 8); в противном случае переходим к этапу 9). Если HOB не может быть изменен, то возвращаем ошибку **updateError** с проблемой **affectsMultipleDSAs**.

   Если DSA имеет NHOB для своего контекста именования с вышестоящим DSA, то способы обнаружения дублирующих статей выходят за рамки данной спецификации Справочника. Переименуйте статью. Если контекст именования имеет подчиненные контексты именования в других DSA, то переходим к этапу 8); в противном случае переходим к этапу 9).

7) DSA должен гарантировать, что новое имя соответствует системной схеме. В противном случае он должен возвратить соответствующую ошибку **attributeError** или **updateError**. DSA должен гарантировать, что не существует других подстатей с именем, которое совпадает с новым именем. В противном случае он должен возвратить ошибку **updateError** с проблемой **entryAlreadyExists**.

8) В соответствующее время DSA должен изменить операционное связывание со всеми релевантными подчиненными DSA, с которыми он имеет иерархические или неспецифические иерархические операционные связывания. Релевантные связывания – это те, которые ассоциируются с контекстами именования, которые являются подчиненными для административной точки, ниже которой располагается статья, для которой изменяется название. Контексты именования, контекстные префиксы которых соответствуют автономным административным точкам, не являются релевантными. Если поддерживается DOP, то следует действовать в соответствии с процедурами из пп. 24.3.2.1 и 25.3.2. Если DOP не поддерживается, то для обновления RHOB следует использовать локальные средства.

9) Если переименованный контекст именования, статья или любой из ее подчиненных, статья псевдонима или подстатья находятся внутри **UnitOfReplication** для одного или более теневых соглашений, которыми обладает данный DSA, то необходимо обновить теневых потребителей с использованием процедур информационной теневой службы Справочника, которые специфицируются в Рек. МСЭ-Т X.525 | ИСО 9594-9.

   Если статья, статья псевдонима или подстатья находится в пределах **UnitOfReplication** для одного или более теневых соглашений, которыми обладает данный DSA, и переименованная статья, статья псевдонима или подстатья не находится в пределах **UnitOfReplication**, то необходимо обновить теневых потребителей с использованием процедур теневых служб Справочника, который специфицируется в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9. В этом случае теневая статья и все ее подчиненные должны быть удалены.

   Если статья, статья псевдонима или подстатья не находится в пределах **UnitOfReplication** для одного или более теневых соглашений, которыми обладает данный DSA, и переименованная статья, статья псевдонима или подстатья теперь находится в пределах **UnitOfReplication**, то необходимо обновить теневых потребителей с использованием процедур теневых служб Справочника, который специфицируется в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9. В этом случае теневая статья и все ее подчиненные должны стать теневыми.

   Если переименованная подчиненная ссылка в непосредственно вышестоящем DSA [чей HOB был изменен на этапе 6)] находится внутри **UnitOfReplication** для одного или более своих теневых соглашений, то необходимо обновить теневых пользователей с использованием процедур информационного теневых служб Справочника, который специфицируется в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

   Если компоненты связывания RHOB с подчиненным DSA [измененного согласно этапу 8) чуть выше] находятся внутри **UnitOfReplication** для одного или более теневых соглашений, относящихся к подчиненному DSA, то необходимо обновить теневых потребителей с использованием процедур информационных теневых служб Справочника, который специфицируется в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

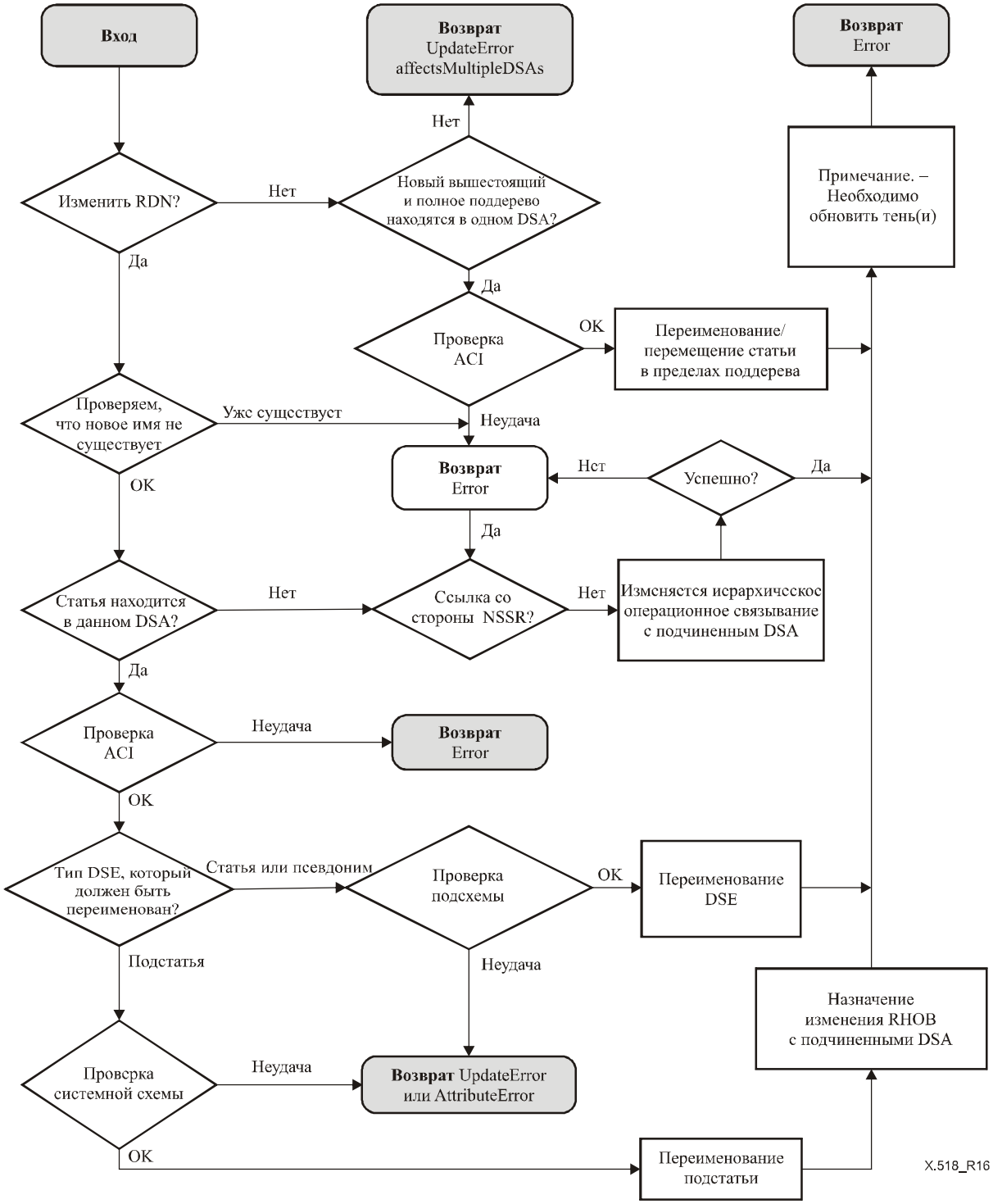


Рисунок 16 – Процедура Modify DN

19.1.5 Операции изменения и неспецифические подчиненные ссылки

Если DSA содержит NSSR и не знает полного набора имен подчиненных для статьи, для которой:

a) была направлена операция **addEntry**; или

b) была направлена операция **modifyDN**,

тогда перед выполнением этой операции DSA может выполнить следующий набор процедур.

1) Если для операции **addEntry** или **modifyDN** установлена опция управления службой **chainingProhibited**, то возвращается ошибка **updateError** с проблемой **affectsMultipleDSAs**.

2) Если DSA не хочет или не может выполнить множественное связывание для исходящих запросов, то возвращается ошибка **serviceError** с проблемой **unwillingToPerform** или **unavailable**, соответственно.

3) DSA должен обеспечить множественное связывание операции **chainedReadEntry** для каждого главного DSA в наборе **accessPointInformation** для NSSR. (DSA должен использовать только главный DSA из каждой **MasterAndShadowAccessPoints** по причине переходного нарушения целостности, которое вызвано теневым копированием). Параметры **ReadArgument** устанавливаются следующими:

   object устанавливается либо как название статьи, которая должна быть добавлена (в случае **addEntry**), или же как предложенное название уже существующей статьи (в случае **modifyDN**).

   selection атрибут класса объекта.

   Параметры **CommonArguments** устанавливаются следующим образом:

   – устанавливается опция управления службой **dontDereferenceAliases**;

   – **OperationProgress.nameResolutionPhase** устанавливается как **completed**.

   Параметры **ChainingArguments** устанавливаются следующим образом:

   – **originator** устанавливается как имя источника;

   – **targetObject** опускается;

   – **OperationProgress.nameResolutionPhase** устанавливается как **proceeding** и **nextRDNToBeResolved** устанавливается равным 1 (количество RDN в имени объекта);

   – **traceInformation** устанавливается как пустая последовательность;

   – **referenceType** устанавливается как **nonSpecificSubordinate**;

   – **timeLimit** устанавливается соответствующим образом в зависимости от входящего запроса.

   Другие параметры, например **SecurityParameters**, также могут устанавливаться соответствующим образом, например, в зависимости от локальной стратегии.

4) DSA ожидает получения всего набора ответов. Если каким то из ответов является **ReadResult**, то должна быть возвращена ошибка как в показанном ниже этапе 6).

5) Если все ответы – это ошибка **serviceError** с проблемой **unableToProceed**, то может выполняться оценка операции.

6) Если возвращается **ReadResult**, то для оригинальной операции следует возвратить **updateError** с проблемой **entryAlreadyExists**;

7) Если в ответ на запрос **readEntry** возвращается какая-то другая ошибка, то необходимо возвратить ошибку **serviceError** с проблемой **unwillingToPerform**.

DSA, которое получило запрос **chainedRead**, должно сообщить ответ в соответствии с наличием или отсутствием статьи и в зависимости от собственной стратегии управления доступом.

19.2 Процедура, которая обращается к единственной статье

Операции **read**, **chainedRead**, **compare** и **chainedCompare** относятся к группе процедур, затрагивающих единственную статью. Такие процедуры содержат только следующие три этапа:

1) Проверка контроля доступа согласно п. 9 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3. Если операция не разрешается, то возвращается соответствующая ошибка безопасности.

2) Выполняем операцию над найденным DSE согласно п. 9 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3.

3) Подготавливаем ответ и возвращаемся.

19.3 Процедура, которая обращается к множеству статей

В зависимости от типа операции запроса (**list** или **search**) следует действовать согласно соответствующей процедуре, которая определяется в пп. 19.3.1 и 19.3.2.

19.3.1 Процедуры List (список)

Данный подраздел специфицирует процедуры оценки, специфические для операций **list** и **chainedList**.

Следует действовать согласно процедуре **List (I)**, когда в запросе List компонент **operationProgress.nameResolutionPhase** установлен как **notStarted** или **proceeding**, и когда DSA после выполнения разрешения имени обнаруживает, что он содержит базовый объект. Процедура **List (II)** должна использоваться в том случае, когда в запросе List компонент **nameResolutionPhase** установлен как **completed**.

19.3.1.1 Параметры процедуры

19.3.1.1.1 Аргументы

Данная процедура использует следующие аргументы:

- аргумент **ListArgument**;
- целевой DSE **e**;
- **operationProgress** для **chainingArgument**.

19.3.1.1.2 Результаты

Если данная процедура выполняется успешно, то она возвращает:

- набор подчиненных для **e** в **listInfo.subordinates**;
- **limitProblem**, который передается в **partialOutcomeQualifier**;
- набор ссылок продолжения в списке **SRcontinuationList**.

19.3.1.2 Определение процедуры

19.3.1.2.1 Процедура List(I)

Процедура **List (I)** состоит из следующих этапов, которые показаны на рис. 17:

1) Если установлено управление службой **subentry**, то переходим к этапу 5); в противном случае переходим к этапу 2).
2) Если DSE **e** относится к типу **nssr**, то добавляем ссылку продолжения в список **SRcontinuationList** со следующими установленными компонентами:
   - **targetObject** установлен равным первичному выделенному имени DSE **e** (в RDN могут включаться альтернативные выделенные значения);
   - **aliasedRDNs** отсутствует;
   - **operationProgress** с **nameResolutionPhase** установлен как **completed** и отсутствует **nextRDNtoBeResolved**;
   - **rdnsResolved** отсутствует;
   - **referenceType** установлен как **nonSpecificSubordinate**;
   - **accessPoints** определена как набор **accessPointInformation**, каждая из которых получена от значения атрибута **nonSpecificKnowledge** для DSE **e**.
3) Для каждого DSE **e'**, которое является непосредственным подчиненным для DSE **e**, выполняются следующие этапы:
   a) Проверяем ACI, которые имеются в **e'**. Если ACI не разрешает представление RDN из **e'** в качестве списка, то пропускаем данное DSE. Если ACI отсутствует (например, в случае подчиненной ссылки и склейки), то решение о том, продолжать или нет обработку, является предметом местной стратегии.
   b) Проверяем все типы DSE для **e'**.
      i) Если **e'** относится к типу **subr**, то возможны два варианта. В первом из них ACI подчиненной статьи и класс объекта доступны локально. В этом случае, основываясь на локальной стратегии и разрешениях ACI добавляем RDN, относящийся к **e'**, к **listInfo.subordinates** со значением **aliasEntry** равным **TRUE**, если **e'** относится к типу **sa**, а **fromEntry** установлен как **FALSE**. Во втором варианте, когда в **e'** не содержит ACI для статьи, добавляем ссылку продолжения к списку **SRcontinuationList**, при этом устанавливаем следующие компоненты:
         - **targetObject** устанавливается равным первичному выделенному имени DSE **e** (альтернативные выделенные значения могут включаться в RDN);
         - **aliasedRDNs** отсутствует;
         - **operationProgress** с **nameResolutionPhase** установлен как **completed** и отсутствует **nextRDNtoBeResolved**;
         - **rdnsResolved** отсутствует;
         - **referenceType** установлен как **subordinate**;
         - **accessPoints** равно значению, содержащемуся в атрибуте **specificKnowledge**, относящемуся к DSE **e'**.
      ii) Если DSE **e'** относится к типу **entry** или **glue**, то добавляем RDN, относящийся к **e'**, к **listInfo.subordinates** с **aliasEntry** равным **FALSE** и **fromEntry**, установленным в соответствии с тем, является ли **e'** копией.

         ПРИМЕЧАНИЕ. – В том случае, если **e'** является **glue**, он должен иметь один или более подчиненных, что подразумевает, что он не может являться псевдонимом в главном DSA. Кроме этого, любой относящийся к операции List (список) ACI хранится в данном DSE, которые поставляется с помощью теневого протокола.

      iii) Если DSE **e'** относится к типу **alias**, то добавляем RDN, относящийся к **e'**, к **listInfo.subordinates**, при этом **aliasEntry** устанавливается как **TRUE**, а **fromEntry** устанавливается в зависимости от того, является ли **e'** копией.

c) Проверяем, не превышены ли ограничения по времени, размеру или административному лимиту. Если это имеет место, то соответственным образом устанавливаем **limitProblem** в **partialOutcomeQualifier** и возвращаемся.

d) Переходим к этапу 3) a) до тех пор, пока не будут обработаны все подчиненные DSE.

4) Если обработаны все нижестоящие DSE, то возвращаемся к диспетчеру операций **Operation Dispatcher**.

5) Для каждой подстатьи **e'**, которая является непосредственно подчиненным для DSE **e**, выполняем следующие этапы:

a) Проверяем ACI в **e'**. Если ACI не разрешает представление в виде списка RDN из **e'**, то пропускаем данный DSE. В противном случае добавляем RDN из **e'** к **listInfo.subordinates**, при этом **aliasEntry** устанавливается равным **FALSE** и **fromEntry** устанавливается в соответствии с тем, является ли **e'** копией.

b) Проверяем, не превышены ли ограничения по времени, размеру или административному лимиту. Если это имеет место, то соответственным образом устанавливаем **limitProblem** в **partialOutcomeQualifier** и возвращаемся.

6) Возвращаемся в диспетчер операций **Operation Dispatcher**.

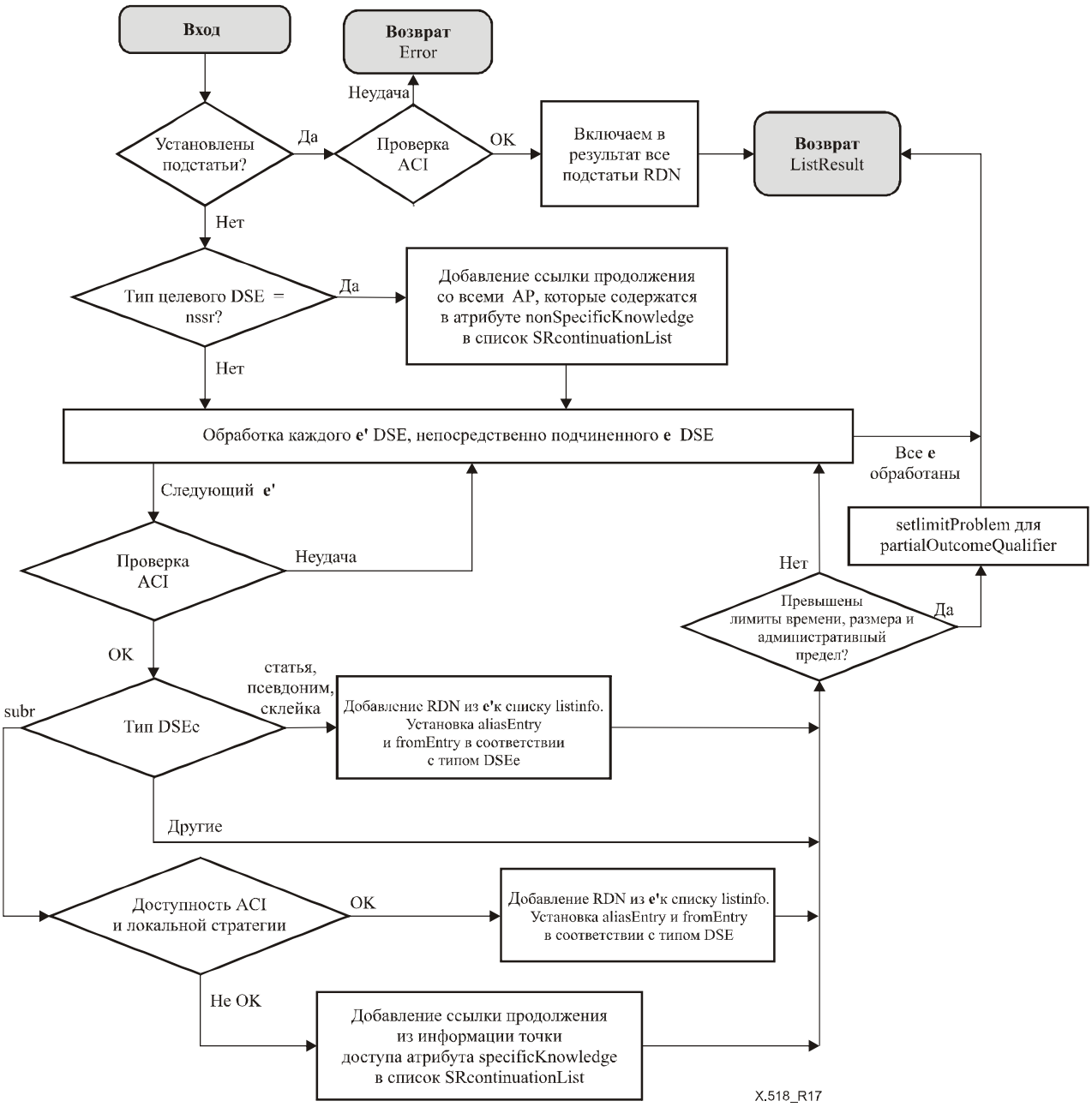


Рисунок 17 – Процедура List (I)

19.3.1.2.2 Процедура List (II)

Процедура **List (II)** состоит из следующих этапов, которые показаны на рис. 18:

1) Для каждого DSE **e'**, который является непосредственным подчиненным для DSE **e**, выполняются этапы от 1), a) до 1), d):
   a) Если **e'** не является статьей или псевдонимом, то переходим к следующему непосредственному подчиненному.
   b) Проверяем ACI для **e'**. Если ACI не разрешает проведение операции, то переходим к следующему непосредственному подчиненному **e**.
   c) Добавляем RDN из DSE **e'** к **listInfo.subordinates**, при этом компонент **aliasEntry** из **listInfo.subordinates** устанавливаем в зависимости от того, является ли **e'** псевдонимом, и компонент **fromEntry** устанавливается в зависимости от того, является ли **e'** копией или нет. Если **excludeShadows** равен **TRUE**, то игнорируем DSE, которые относятся к типу **shadow** или **writableCopy**.
   d) Проверяем, не превышены ли ограничения по времени, размеру или административному лимиту. Если это имеет место, то соответствующим образом устанавливаем **limitProblem** в **partialOutcomeQualifier** и возвращаемся.
   e) Если обработаны не все подчиненные DSE, то переходим к этапу 1) a).
2) Если обработаны все подчиненные DSE, то проверяем, от кого поступил подзапрос – от DAP или от DSP. В том случае, если подзапрос был направлен с помощью DAP, и **ListResult** пуст, то возвращаем диспетчеру операций ошибку **serviceError** с проблемой **invalidReference**. В противном случае возвращаем **ListResult**.

ПРИМЕЧАНИЕ. – **invalidReference** используется как мера предосторожности в том случае, если пользователь не имеет доступа к вышестоящей статье. Если ACI вышестоящей статьи доступен (предоставляется RHOB), то если это разрешено, может возвращаться нулевой результат.

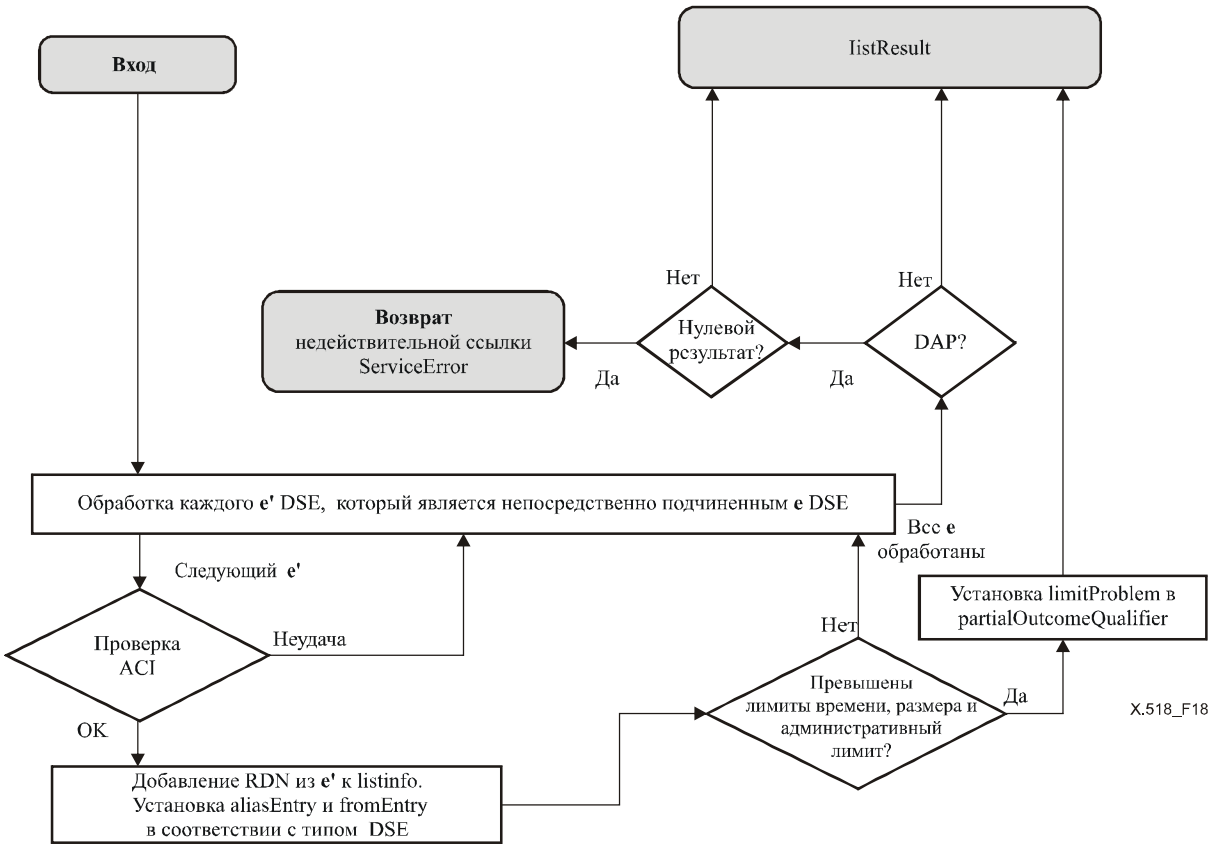


Рисунок 18 – Процедура List (II)

19.3.2 Процедуры Search (поиск)

В данном подразделе специфицируются процедура оценки, которая является специфической для операций **search** и **chainedSearch**.

Следует использовать процедуру **Search-rule-check (I)** в том случае, когда в запросе поиска компонент **operationProgress.nameResolutionPhase** установлен как **notStarted** или как **proceeding**, и когда DSA после выполнения разрешения имени обнаруживает что он содержит целевой объект. Если данная процедура возвращает ошибку, то возвращаемся с этой ошибкой. В противном случае переходим к процедуре **Search (I)**.

Следует использовать процедуру **Search-rule-check (II)** в том случае, когда в запросе **search** компонент **nameResolutionPhase** установлен как **completed**. Если эта процедура возвращает ошибку, то возвращаемся с этой ошибкой. В противном случае переходим к процедуре **Search (II)**.

ПРИМЕЧАНИЕ. – Если **nameResolutionPhase** равно **completed**, то ожидается, что целевой объект является непосредственным подчиненным контекстного префикса.

19.3.2.1 Параметры процедуры

19.3.2.1.1 Аргументы

Аргументы, которые используются в данной процедуре:

- аргумент **SearchArgument**;
- целевой DSE **e**;
- **operationProgress** из **ChainingArguments**;
- **exclusions** из **ChainingArguments** (список тех RDN, которые необходимо исключить из поиска);
- **traceInformation** из **ChainingArguments**;
- **searchRuleId** из **ChainingArguments**;
- **chainedRelaxation** из **ChainingArguments**; и
- **relatedEntry** из **ChainingArguments**.

19.3.2.1.2 Результаты

Если процедура успешно выполнена, то она возвращает:

- набор статей, для которых обнаружено сопоставление, в **searchResult.entryInformation**;
- **alreadySearched** в **ChainingResults**;
- в зависимости от условий, количество в **partialOutcomeQualifier.entryCount**; и
- набор ссылок продолжения в **SRcontinuationList**.

19.3.2.2 Определение процедуры

19.3.2.2.1 Процедура относящегося аргумента статьи (Related Entry Argument)

Данная процедура имеет отношение только тогда, если в запросе **search** имеется компонент **joinArguments** и **ChainingArguments** (если имеется) не содержит компонент **relatedEntry**.

1) Если запрос **search** является защищенным, то генерируется DSP-запрос для каждого элемента компонента **joinArguments**, каждый из них содержит оригинальный запрос DAP или же LDAPMessage. Значения **ChainingArguments** устанавливаются следующим образом:
   - если входящий запрос содержит **ChainingArguments** с компонентом **originator**, то значение данного компонента копируется в компонент **originator** сгенерированных запросов; в противном случае использование данного компонента определяется локальной политикой безопасности;

     ПРИМЕЧАНИЕ. – Может случиться, что принимающий DSA не умеет использовать название, указанное в данном компоненте, так как оно исходит от отличного DIT.
   - компонент **operationProgress** должен опускаться или же ему присваивается значение по умолчанию;
   - компоненты **traceInformation**, **aliasDereferenced**, **aliasedRDNs**, **returnCrossRefs**, **entryOnly**, **exclusions**, **nameResolutionOnMaster**, **searchRuleId**, **chainedRelaxation** опускаются; и
   - компонент **relatedEntry** устанавливается равным значению, которое соответствует относительной позиции **JoinArgument**, который применяется к DSA, которому передается данный запрос; если первому **JoinArgument** присвоено значение 0, то следующему присваивается значение 1 и т. д.
2) Если входящий запрос не защищен, то генерируется DSP запрос для каждого элемента компонента **jointArguments**, при этом аргумент **SearchArgument** создается следующим образом:
   - компонент **baseObject** должен копироваться из компонента **joinBaseObject** соответствующего **JoinArgument**;
   - компонент **subset** должен копироваться из компонента **joinSubset** соответствующего **JoinArgument**;
   - компонент **filter** должен копироваться из компонента **filter** соответствующего **JoinArgument**; и

– остающиеся компоненты должны оставаться теми же, что и в оригинальном запросе, за исключением компонентов **joinArguments** и **joinType**, которые опускаются.

Значения **ChainingArguments** должны устанавливаться аналогично обсуждавшемуся выше защищенному запросу, за тем исключением, что компонент **relatedEntry** опускается.

3) Вызывается **Operation Dispatcher** для каждого запроса с целью его локального продолжения.
4) Если **Operation Dispatcher** возвращает ошибку **referral** или ошибки, связанные с занятостью или отсутствием, тогда добавляем (или же создаем и добавляем) ссылку продолжения к **partialOutcomeQualifier** из **SearchResult** и после этого возвращаемся.
5) Если **Operation Dispatcher** возвращает другие ошибки, то отбрасываем их и возвращаемся.
6) Если **Operation Dispatcher** возвращает **SearchResult**, тогда:
   i) Если результат подписан (снабжен подписью), зашифрован или же подписан и зашифрован, тогда добавляем его к **uncorrelatedSearchInfo** из **SearchResult**.
   ii) Если результат не подписан, не зашифрован и не подписан и зашифрован одновременно, то выполняем процесс объединения согласно Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3.

19.3.2.2.2 Процедуры проверки правил поиска (I)

Эта процедура имеет отношение только тогда, когда DSA поддерживает специфические для службы административные области.

Если компонент **searchRuleId** присутствует в **ChainingArguments**, то операция является результатом процедуры разыменования псевдонима, которая произошла во время предыдущей фазы оценки. В этом случае, если целевой DSE находится внутри специфической для службы административной области с отличающимся **dmdId**, или же целевой DSE находится вне специфической для службы административной области, то возвращаем ошибку службы **unwillingToPerform**. В противном случае, выбираем соответствующее правило поиска, основываясь на информации в **searchRuleId**, и возвращаемся.

ПРИМЕЧАНИЕ 1. – Администрация службы была определена как критическое расширение. Когда DSA, который не поддерживает администрацию службы, получает связанный запрос поиска с компонентом **searchRuleId**, он возвращает ошибку **serviceError** с проблемой **unavailableCriticalExtension**.

Если **searchRuleId** отсутствует и целевой DSE находится за пределами специфической для службы административной области или же находится в пределах такой области, однако не существует подстатей, которые ассоциируются с данной областью, то в этом случае возвращаемся.

Если целевой DSE находится в пределах специфической для службы административной области и информация в **traceInformation** сообщает о том, что уже существовала фаза оценки, то возвращаем ошибку службы **unwillingToPerform**.

ПРИМЕЧАНИЕ 2. – Этот такая ситуация, когда поиск начал свою начальную оценку из-за пределов специфической для службы административной области и теперь пытается расширить работу в административную область, специфическую для других служб.

В противном случае следуем такой процедуре:

1) Определяем все правила поиска, которые связаны с целевым DSE, т. е. все правила поиска в подстатьях службы, которые в своих спецификациях поддерева содержат целевой DSE (например, с использованием операционного атрибута **searchRulesSubentry**). Эти правила поиска далее будут называться *правила поиска кандидата* (*candidate-search-rules*). Если таких правил поиска не существует, то создаем ошибку службы с проблемой **requestedServiceNotAvailable**, в **CommonResults** включаем в компонент **notification** атрибут **searchServiceProblem**, которому присвоено значение **id-pr-unidentifiedOperation**, и после этого возвращаемся.
2) Если в запрос поиска включаются управление службой **serviceType** и/или **userClass**, то удаляем все правила поиска, которые не совместимы с этими управлениями службы из правил поиска кандидата. Если после этого список остается пустым, то создаем ошибку службы с проблемой **requestedServiceNotAvailable**; включаем в компонент **notification** из **CommonResults** следующую информацию и затем возвращаемся:
   – атрибут **searchServiceProblem** со значением **id-pr-unidentifiedOperation**;
   – если в запрос поиска включено управление службой **serviceType**, то атрибуту **serviceType** присваиваем значение данного управления службой.
3) Разделяем список правил поиска кандидата на четыре списка (некоторые из них могут являться пустыми):
   – список **GoodPermittedSR** содержит все правила поиска кандидатов, которые инициатор запроса имеет разрешение вызывать и с которыми совместим запрос поиска в соответствии с процедурой проверки корректности поиска, которая специфицирована в п. 13 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3;

     ПРИМЕЧАНИЕ 3. – Если этот список не пуст, то нет необходимости создавать другие списки.

   – список **MatchProblemSR** содержит все правила поиска кандидата, относительно которых инициатор запроса обладает разрешением на вызов и с которыми совместим запрос поиска, за исключением matchingUse для одного или более профилей атрибутов запроса;
   – список **BadPermittedSR** содержит все правила поиска кандидатов, относительно которых инициатор запроса обладает разрешением на вызов, но с которыми не совместим запрос поиска;

– список **DeniedSR** содержит все правила поиска кандидатов, относительно которых инициатор запроса не обладает разрешениями на вызов.

4) Если список **GoodPermittedSR** содержит одно или более пустых правил поиска, то с помощью локального алгоритма выбираем одно из пустых правил поиска в качестве управляющего правила поиска и затем возвращаемся.

5) Если список **GoodPermittedSR** не является пустым, то отбрасываем все правила поиска за исключением того, которое имеет наибольший показатель для **userClass**.

6) В списке **GoodPermittedSR** с помощью локального алгоритма выбираем одно из остающихся правил поиска в качестве управляющего правила поиска и возвращаемся.

ПРИМЕЧАНИЕ 4. – Если в этом списке существует несколько правил поиска, из которых можно выбирать, то реализация должна зафиксировать это для администратора, так как вероятно понадобится переработка правил поиска.

7) Если список **MatchProblemSR** не является пустым, то действуя аналогично алгоритмам, описанным в 5) и 6), выбираем одно из правил поиска, генерируем ошибку службы и связанную с ней информацию согласно п. 13.4 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3, и затем возвращаемся.

8) Если список **DeniedSR** пуст, то переходим к 10), в противном случае отбрасываем из списка все правила поиска, с которыми не согласовывается запрос поиска, и отбрасываем каждое пустое правило поиска. Если список теперь становится пустым, то переходим к 10); в противном случае генерируем ошибку службы с проблемой **requestedServiceNotAvailable**; включаем в компонент **notification** из **CommonResults** описанные ниже подкомпоненты и возвращаемся:
   – атрибуту **searchServiceProblem** присваивается значение **id-pr-unavailableOperation**;
   – если все остающиеся в списке **DeniedSR** правила поиска имеют одинаковые значения для компонента **serviceType**, то присваиваем атрибуту **serviceType** данное значение.

9) Если список **BadPermittedSR** пуст, генерируем ошибку службы с проблемой **requestedServiceNotAvailable**, включаем в компонент **notification** из **CommonResults** следующие подкомпоненты и возвращаемся:
   – атрибут **searchServiceProblem** со значением **id-pr-unidentifiedOperation**.

10) Для каждого нумерованного пункта в процедуре из п. 13.1 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3, которые берутся по порядку, проверяем запрос поиска относительно остающихся правил поиска в списке **BadPermittedSR**, а затем для каждого из этих пунктов:
   – если поиск согласовывается с пунктом для некоторого правила поиска, но не для всех правил поиска, отбрасываем те правила поиска, с которыми не существует согласования;
   – если теперь **BadPermittedSR** содержит только одно правило поиска, выполняем процедуру, специфицированную в п. 13 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3, и возвращаемся;
   – в противном случае проверяем следующий пункт.

11) Если **BadPermittedSR** теперь содержит только правила поиска, с которыми поиск не является согласованным в соответствии с существующей процедурой, то генерируем ошибку службы с проблемой **requestedServiceNotAvailable**; включаем в компонент **notification** из **CommonResults** показанные ниже компоненты и затем возвращаемся:
   – атрибут **searchServiceProblem** со значением **id-pr-unidentifiedOperation**;
   – если все правила поиска в **BadPermittedSR** специфицируют один и тот же тип службы, атрибут **serviceType** получает этот тип службы как свое значение.

12) Для каждого последовательно взятого нумерованного пункта из п. 13.2 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3 проверяем запрос поиска относительно остающихся в **BadPermittedSR** правил поиска и для каждого пункта:
   – если запрос согласовывается с пунктом для некоторых, но не для всех правил поиска, отбрасываем те правила поиска, с которыми нет согласования;
   – если теперь **BadPermittedSR** содержит только одно правило поиска, то выполняем процедуру, специфицированную в п. 13 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3, и затем возвращаемся;
   – в противном случае, проверяем следующий пункт.

13) Для каждого последовательно взятого нумерованного пункта из п. 13.3 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3 проверяем запрос поиска относительно остающихся в **BadPermittedSR** правил поиска и затем для каждого пункта:
   – если поиск согласовывается с пунктом для некоторых, но не для всех правил поиска, отбрасываем те правила поиска, с которыми нет согласования;
   – если теперь **BadPermittedSR** содержит только одно правило поиска, то выполняем процедуру, специфицированную в п. 13 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3, и затем возвращаемся;
   – в противном случае, проверяем следующий пункт.

14) Генерируем ошибку службы с проблемой **requestedServiceNotAvailable**; включаем в компонент **notification** из **CommonResults** описанные ниже подкомпоненты и затем возвращаемся:
   – атрибут **searchServiceProblem** со значением **id-pr-unidentifiedOperation**;
   – если все правила поиска в **BadPermittedSR** специфицируют один и тот же тип службы, атрибут **serviceType** получает этот тип службы как свое значение.

19.3.2.2.3 Процедура проверки правил поиска (II)

Данная процедура уместна только тогда, когда DSA поддерживает специфические для административные области.

Если отсутствует **searchRuleId** и все непосредственные подчиненные статьи (контекстные префиксы) целевого DSE являются специфическими для службы административными точками, то возвращаем ошибку **serviceError** с проблемой **unwillingToPerform**. Если же некоторые из подчиненных статей не являются специфическими для службы административными точками, то выбираем соответствующие контексты именования для оценки поиска и затем возвращаемся.

Если же **searchRuleId** присутствует, то проверяется каждая подчиненная статья целевого DSE с целью проверить, что она находится в пределах той же специфической для службы административной области, что и целевой DSE. Если это не так, то соответствующий контекст именования исключается из поиска. Если остаются контексты именования (включая те, что относятся к выполняющему DSA), в которых поиск может быть продолжен, выбираем указанное в **searchRuleId** правило поиска и возвращаемся. Если не остается контекстов именования, в которых поиск может быть продолжен, генерируем **serviceError** с проблемой **unwillingToPerform** и возвращаемся.

ПРИМЕЧАНИЕ. – Последнее не должно происходить, если информация знания совместима между данным DSA и тем DSA, который содержит вышестоящий контекст именования.

19.3.2.2.4 Выбор информации статьи

Для сопоставленных статей и для статьи, выбранной как часть выбора иерархии, информация атрибута выбирается как пересечение следующего:

a) того, что специфицируется с помощью **searchArgument.selection**, возможно измененного с помощью спецификаций контекста по умолчанию, а для сопоставленных статей также с помощью **searchArgument.matchedValuesOnly**;
b) того, что определяется управляющим правилом поиска (если оно существует).

Данная информация статьи добавляется к списку статей в **searchResult.entryInformation**.

Добавляются только те атрибуты, чей размер (тип и все значения) не превосходит **attributeSizeLimit**.

19.3.2.2.5 Процедура Search (I)

Это рекурсивная процедура, которая применяется к запросу **search**, который начинается с данной целевой записи **e**. Она ищет целевую статью **e** и затем обрабатывает DSE, которые являются непосредственно подчиненными для **e**. Если необходимо провести поиск для всего поддерева, то данная процедура вызывает сама себя рекурсивно. Процедура состоит из следующих этапов и показана на рис. 19:

1) Если типом DSE **e** является тип **cp** (DSE в контекстном префиксе), то проверяем каждый элемент аргумента **exclusions** на то, является ли он префиксом DN для **e**.
   a) Если да, то возвращаемся.
   b) В противном случае вызываем Check Suitability (проверка соответствия).
      i) Если **e** не является соответствующим, то создаем **continuationReference** показанным ниже способом и добавляем ее к списку **SRContinuationList**:
         – **targetObject** устанавливается как DN для DSE **e**;
         – **operationProgress** с **nameResolutionPhase** устанавливаем как **proceeding** и **nextRDNtoBeResolved** устанавливаем равным количеству RDN в **e**;
         – все остальные компоненты **continuationReference** остаются неизменными.

         После этого возвращаемся.

         ПРИМЕЧАНИЕ 1. – Это единственное место, где подзапрос **search** связывается с теневым поставщиком. В других словах, целевой объект для такого связанного подзапроса всегда является контекстным префиксом.

      ii) В противном случае добавляем выделенное имя **e** к **alreadySearched** из **ChainingResults**.

         ПРИМЕЧАНИЕ 2. – **alreadySearched** содержит исключительно контекстные префиксы.
2) Если **e** относится к типу **alias** и **searchAliases** в **SearchArgument** равен **TRUE**, тогда вызываем процедуру **Search Alias** и после этого возвращаемся.
3) Если **subset** равен **oneLevel**, то переходим к этапу 6).

   ПРИМЕЧАНИЕ 3. – В данной точке **e** не может являться неполным подчиненным, так как в контекстном префиксе проверка соответствия (Check Suitability) должна была гарантировать, что такое не должно произойти.
4) Если **subset** является **baseObject**, или же если **entryOnly** равен **TRUE**, то продолжаем работать в данном этапе, в противном случае переходим к этапу 5).

Если верно одно из следующих:

a) **e** относится к типу **subentry** и установлен контроль службы **subentry**; или

b) **e** не относится к типу **subentry** и контроль службы **subentry** не установлен, то выполняем следующие этапы:

i) Проверяем ACI. Если операция запрещена, то возвращаемся.

ii) Применяем к DSE **e** аргумент фильтра, определенный в **SearchArgument.filter**. Проверяем, что разрешен доступ ко всем атрибутам, которые используются в фильтре, как это определено в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если фильтр совпадает и если статья не была исключена при выборе иерархии, то добавляем информацию атрибута как это определено в п. 19.3.2.2.3.

iii) Если контроль поиска **hierarchySelection** включен в запрос **search** (возможно, подвергнувшись изменениям со стороны спецификации правила поиска), статья является частью иерархической группы, которая имеет более чем одного члена, и установлен индикатор более чем **self**, тогда вызываем процедуру **Hierarchy Selection (I)**.

После этого возвращаемся.

5) Если **subset** равен **subtree** (и **entryOnly** не равен **TRUE**), и в дополнение к этому выполняется одно из следующих условий:

a) **e** относится к типу **subentry** и установлено управление службой **subentry**; или

b) **e** не относится к типу **subentry** и управление службой **subentry** не установлено, то выполняем следующие этапы:

i) Проверяем ACI. Если операция не разрешена, то переходим к этапу 6).

ii) Применяем к DSE **e** аргумент фильтра, определенный в **SearchArgument.filter**. Проверяем, что разрешен доступ ко всем атрибутам, которые используются в фильтре, как это определено в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2. Если фильтр совпадает и если статья не была исключена при выборе иерархии, то добавляем информацию атрибута как это определено в п. 19.3.2.2.3.

iii) Если контроль поиска **hierarchySelection** включен в запрос **search** (возможно, подвергнувшись изменениям со стороны спецификации правила поиска), запись является частью иерархической группы, которая имеет более чем одного члена, и установлен индикатор более чем **self**, тогда вызываем процедуру **Hierarchy Selection (I)**.

iv) Переходим к этапу 6).

6) Если **e** относится к типу **nssr**, то добавляем к списку **SRcontinuationList** ссылку продолжения со следующими компонентами:

– **targetObject** к первичному выделенному имени DSE **e** (альтернативные выделенные значения могут включаться в RDN);

– **aliasedRDNs** отсутствует;

– **operationProgress** с значением **nameResolutionPhase**, установленным как **completed**, и отсутствующим **nextRDNtoBeResolved**;

– **rdnsResolved** отсутствует;

– **referenceType** установлен как **nssr**;

– **accessPoints** установлен как **AccessPointInformation**, полученный от значения(ий) обнаруженного(ых) в атрибуте **nonSpecificKnowledge**.

7) Обрабатываем все DSEs **e'**, которые располагаются как непосредственно подчиненные целевого DSE **e** до тех пор, пока не будут обработаны все подчиненные DSE. Если **e** находится внутри специфической для службы административной области, то должны обрабатываться только те непосредственно подчиненные DSE, которые также являются частью этой специфической для службы административной области. Если **e** находится вне специфической для службы административной области, то не должны обрабатываться те непосредственно подчиненные DSE, которые являются частью специфической для службы административной области. При выполнении этого цикла в том случае, если список сопоставленных статей в **searchResult.entryInformation** превышает ограничение по размеру, времени или вступает в силу административный лимит, то устанавливаем соответствующим образом **limitProblem** в **partialOutcomeQualifier** и затем возвращаемся.

ПРИМЕЧАНИЕ 4. – Проверка на ограничение по размеру также неявно проводится каждый раз при обновлении **searchResult**.

a) Если DSE **e'** относится к типу **subr**, не относится к типу **cp** и не представляет собой подчиненную запись, которая является специфической для службы административной точкой, то добавляем ссылку продолжения к списку **SRcontinuationList** со следующим компонентами:

– **targetObject** устанавливается равным первичному выделенному имени DSE **e** (альтернативные выделенные значения могут включаться в RDN);

– **aliasedRDNs** отсутствует;

– **operationProgress** с **nameResolutionPhase** устанавливается как **completed**, **nextRDNtoBeResolved** отсутствует;

– **rdnsResolved** отсутствует;

– **referenceType** устанавливается как **subr**;

– **accessPoints** устанавливается согласно информации о точке доступа, которая содержится в DSE **e'** в атрибуте **specificKnowledge**.

ПРИМЕЧАНИЕ 5. – Если **e'** относится как к типу **cp**, так и к типу **subr**, то потенциально подзапрос поиска может быть сгенерирован либо подчиненной ссылкой, либо знанием поставщика, но ни в коем случае обоими одновременно. Данная процедура использует второй вариант (ссылки поставщика находятся в cp).

b) Для всех случае:

i) Если **subset** равен **oneLevel**, устанавливаем **entryOnly** равным **TRUE**.

ii) Рекурсивно выполняем процедуру **Search (I)** для целевого DSE **e'**.

8) Если были обработаны все подчиненные, то возвращаемся к диспетчеру операций для последующей обработки.

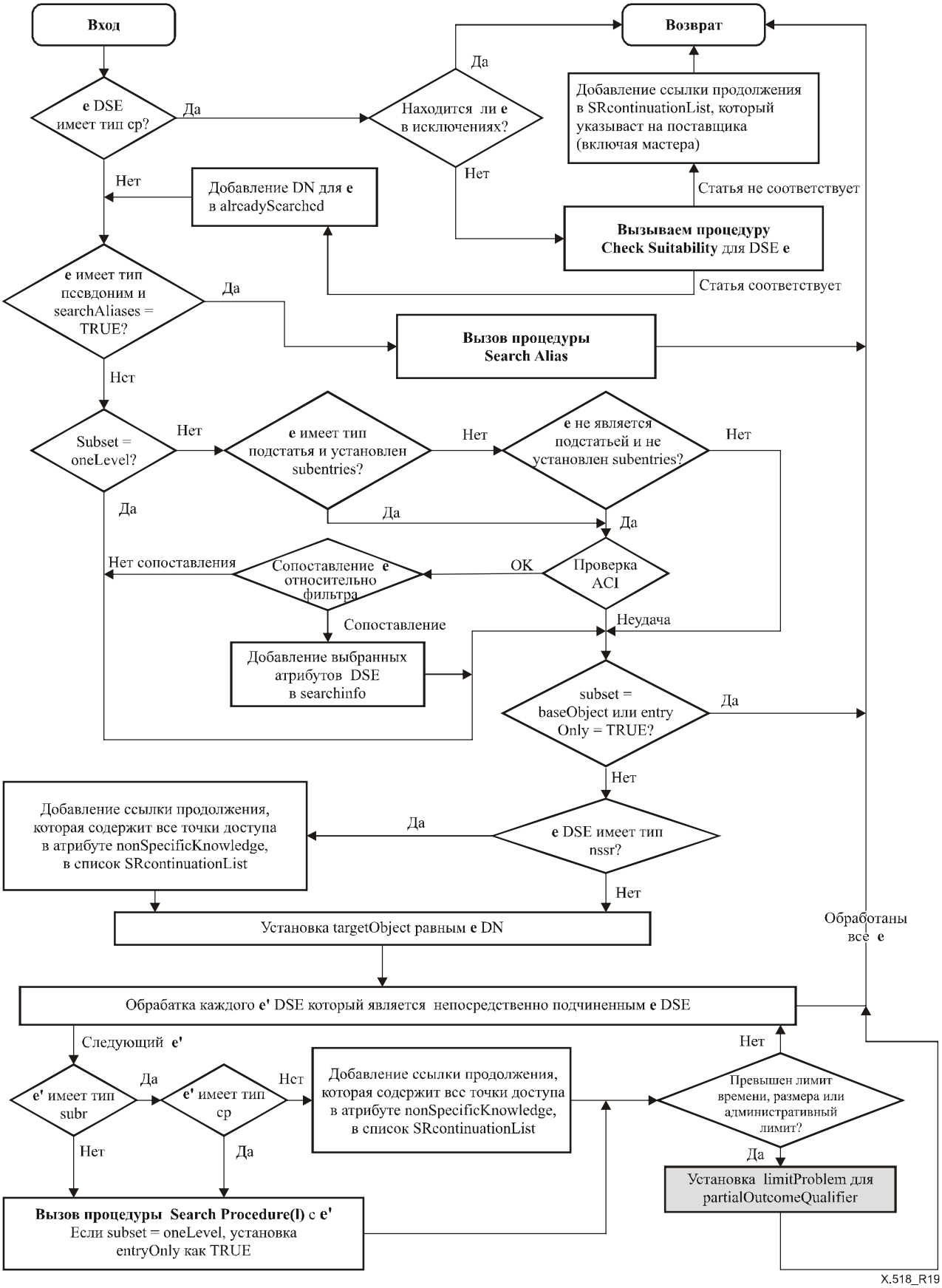


Рисунок 19 – Процедура Search (I)

19.3.2.2.6 Процедура Search (II)

Данная процедура применяется в том случае, если обрабатывается запрос поиска, который возник при декомпозиции запроса в DSA, от которого был получен запрос. Процедура обрабатывает DSE, расположенные ниже целевой DSE **e**, и вызывает процедуру **Search (I)** для каждой статьи объекта:

1) Обрабатываем все DSE **e'**, которые располагаются в качестве непосредственных подчиненных для целевого DSE **e** пока не будут обработаны все подчиненные DSE. После того, как обработаны все подчиненные, возвращаемся в диспетчеру операций для последующей обработки.
2) Если DSE не относится к типу **cp**, то игнорируем его. Возвращаемся к этапу 1).
3) Вызываем **Check Suitability (**проверка соответствия**)**. Если соответствует, то переходим к этапу 4); в противном случае игнорируем и возвращаемся к этапу 1).
4) Выполняем для DSE **e'** процедуру поиска **Search Procedure (I)** согласно п. 19.3.2.2. Если DSE относится к типу **alias** и значение параметра **subset** установлено как **oneLevel**, то устанавливаем **ChainingArguments.entryOnly** равным **TRUE** при вызове процедуры **Search (I)**. Возвращаемся к этапу 1).

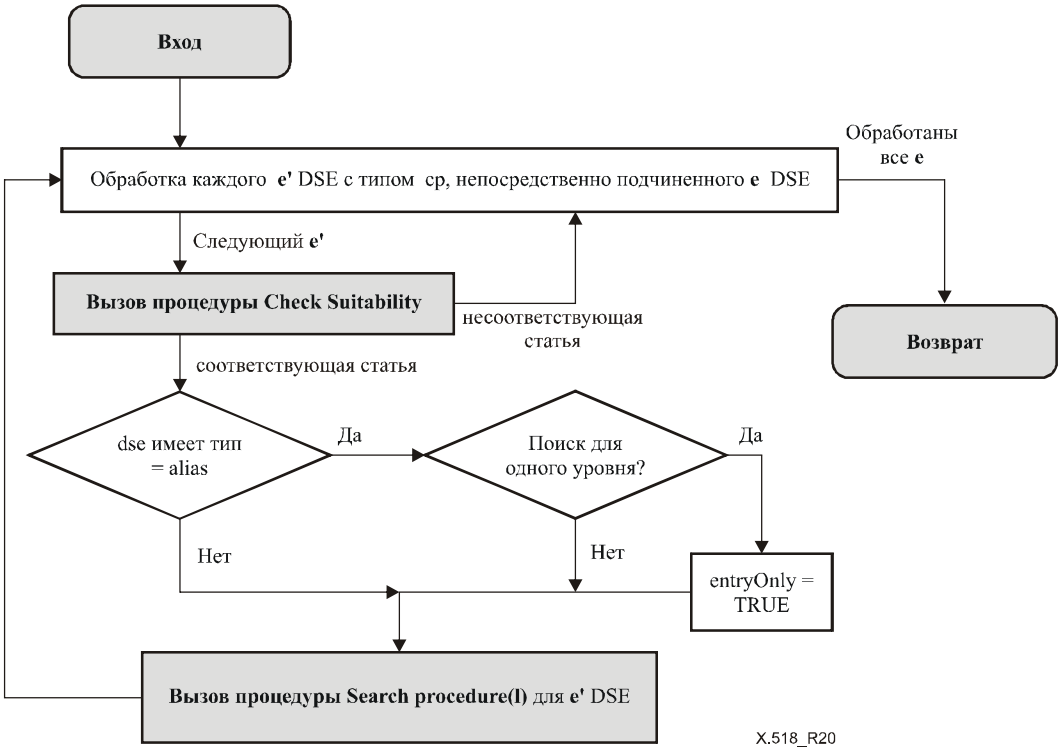


Рисунок 20 – Процедура Search (II)

19.3.2.2.7 Процедура Search Alias (поиск псевдонима)

Данная процедура выполняется в том случае, если во время обработки запроса **search** был обнаружен DSE, который относится к типу **alias** (см. рис. 21):

1) Если **subset** равняется **baseObject** или **oneLevel**, то переходим к этапу 4).
2) Если **aliasedEntryName** является префиксом для **targetObject** или для **baseObject** или же для любого из предыдущих значений **targetObject** в **ChainingArguments.traceInformation**, то псевдоним исключается из поиска по той причине, что это вызовет рекурсивный поиск с дублированием результатов.
3) Если **targetObject**, или **baseObject**, или же любое из предыдущих значений **targetObject** в **ChainingArguments.traceInformation** является префиксом для **aliasedEntryName**, то не требуется осуществлять какую-то специфическую обработку псевдонима по той причине, что для поддерева с данным псевдонимом в любом случае будет производиться поиск.

ПРИМЕЧАНИЕ. – Для каждого из указанных выше случаев **baseObject** не может являться префиксом **targetObject** по причине разыменования псевдонима.

4) Если поиск выполняется внутри специфической для службы административной области и если специфическая для службы административная точка не является префиксом для **aliasedEntryName**, то не требуется осуществлять какую-то специфическую обработку, так как статья с псевдонимом находится вне специфической для данного службы административной области.
5) Создаем **DSP**-запрос с **targetObject**, установленным как **aliasedEntryName**. Если **subset** установлен как **oneLevel**, то задаем **entryOnly** равным **TRUE**. Вызываем **Operation Dispatcher** для локального продолжения запроса.

6) Если **Operation Dispatcher** возвращает ошибку **referral**, или ошибки занятости или отсутствия, то добавляем (или же создаем и добавляем) ссылку продолжения к **partialOutcomeQualifier** из **SearchResult**, а затем возвращаемся.
7) Если **Operation Dispatcher** возвращает другие ошибки, то отбрасываем их и затем возвращаемся.
8) Если **Operation Dispatcher** возвращает **SearchResult**, тогда:
   i) Если результат снабжен подписью, зашифрован или же снабжен подписью и зашифрован одновременно, то добавляем его к **uncorrelatedSearchInfo** из **SearchResult**.
   ii) Если результат не снабжен подписью, не зашифрован или же не снабжен подписью и не зашифрован одновременно, то добавляем его к **searchInfo** из **SearchResult**.

   После этого возвращаемся.

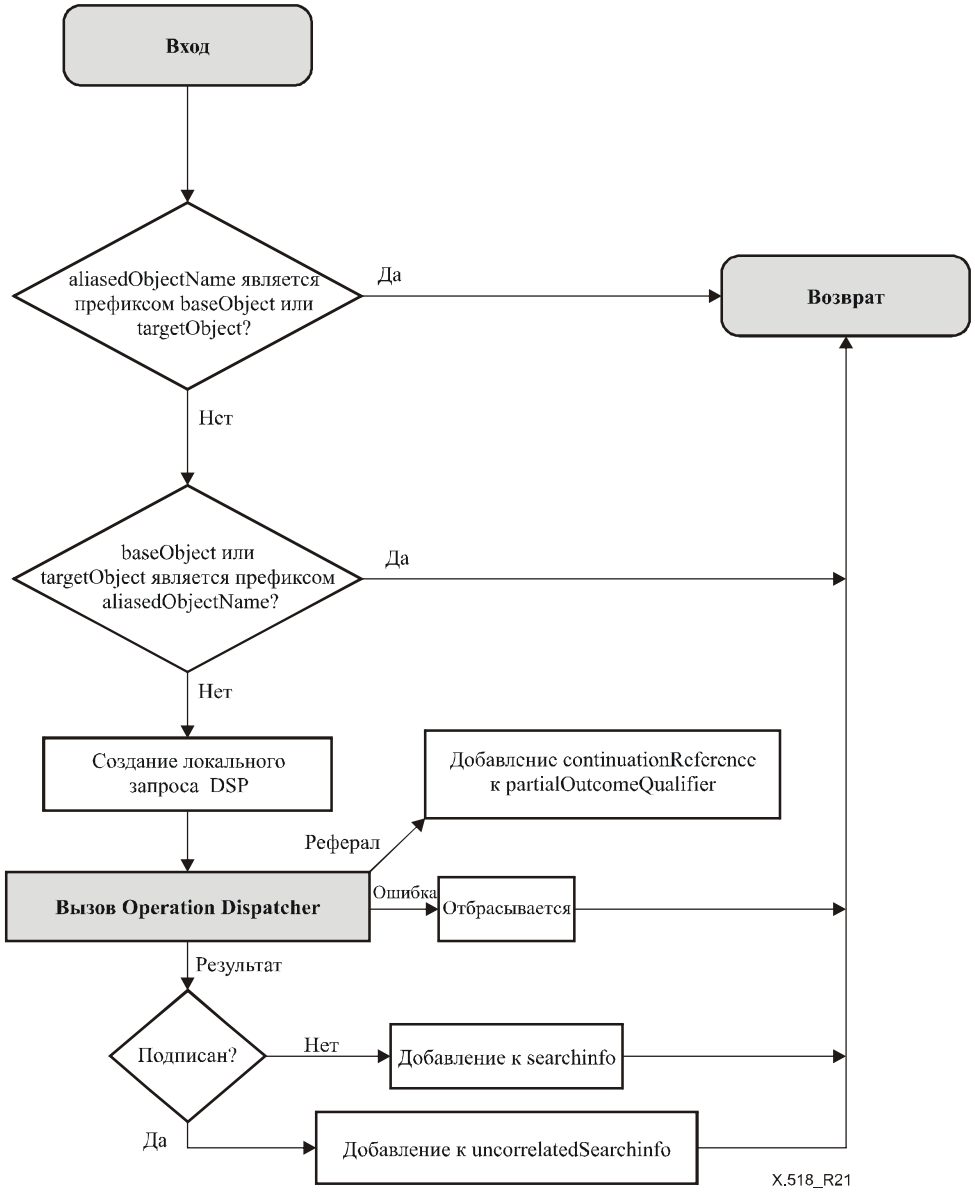


Рисунок 21 – Процедура Search Alias

19.3.2.2.8 Процедура Hierarchy Selection (I) (выбор иерархии)

Процедура выполняется, если при обработке запроса поиска, который специфицирует иерархический выбор, обнаруживается член иерархической группы.

а) Если присутствует иерархический выбор, который не поддерживается DSA, то возвращаемся со следующими условиями:
   – **serviceError** с проблемой **requestedServiceNotAvailable**;
   – атрибутом извещения **searchServiceProblem** с установленным значением **id-pr-unavailableHierarchySelect**;

- атрибутом извещения **serviceType** с установленным значением, равным компоненту **serviceType** из правила поиска; и
- атрибутом извещения **hierarchySelectList**, сообщающем о неправильном выборе.

b) В противном случае добавляем все статьи, которые определены иерархическим выбором согласно п. 19.3.2.2.4. Если это приводит к тому, что не было добавлено ни одной статьи, т. е. иерархический выбор определяет только те статьи, которые не существуют, то устанавливаем глобальную переменную **emptyHierarchySelect**.

20 Процедуры Continuation Reference (ссылки продолжения)

Процедуры в данном разделе вызываются для обработки списка ссылок продолжения (**NRcontinuationList** или **SRcontinuationList**), который создается другими процедурами.

Процедуры **Continuation Reference** состоят из этапов, показанных на рис. 24, 25 и 26. Первым этапом является определение наборов ссылок продолжения из списка продолжения, у которых компонент целевого объекта является одинаковым. Они создаются из набора подчиненных или неспецифических подчиненных ссылок, связанных с одной и той же статьей в DIT. В каждом из этих наборов могут находиться ссылки продолжения, которые встречаются более одного раза. Необходимо просмотреть такие наборы и отбросить все обнаруженные дубликаты.

Эти наборы (каждый содержит свой отличный компонент **targetObject**) могут обрабатываться, последовательно или же параллельно с помощью DSA, так как отсутствует риск получить одни и те же результаты от любого из пары различных наборов. Тем не менее, необходимо контролировать обработку каждой ссылки продолжения в пределах какого-то набора и для каждой **AccessPointInformation** в данной ссылке продолжения, а также для каждой точки доступа в пределах какой-то **AccessPointInformation**, в противном случае, как это описывается в п. 20.1, могут возникнуть дублирующие результаты.

Процедура, которая принята в процедуре **APInfo**, по отдельности обрабатывает один за другим наборы точек доступа, которые содержатся в одном **AccessPointInformation**. Все они указывают (являются копиями) одного контекста именования (или же набора контекстов именования, который для случая NSSR содержится в одном DSA). Если первая точка доступа приводит к результату или тяжелой ошибке, то нет необходимости обрабатывать следующие точки доступа. Если же ошибка является мягкой ошибкой, т. е. **serviceError** (с проблемой **busy**, **unavailable**, **unwillingToPerform**, **invalidReference** или **administrativeLimitExceeded**), то DSA в качестве локальной опции может выбрать обработку другой точки доступа из данного набора.

Обработка значений **AccessPointInformation** в пределах данного набора ссылок продолжения осуществляется однородным способом, без учета того, от какой именно ссылки продолжения они произошли. (По той причине, что два DSE типа **subr**, располагающиеся ниже одной и той же статьи, создают две ссылки продолжения, каждая из которых содержит одно значение **AccessPointInformation**, в то время как один DSE типа **nssr** для двух одинаковых подчиненных (предполагается, что они содержатся в различных DSA) создает одну ссылку продолжения, которая содержит набор из двух значений **AccessPointInformation**.)

Значения **accessPointInformation** могут обрабатываться последовательно или параллельно, как это описывается в п. 20.1. Параллельная стратегия с большей вероятностью способна создать дублирующие результаты. Дублирующие результаты всегда должны быть отброшены.

20.1 Стратегия связывания в присутствии теневого копирования

При наличии теневого копирования, когда необходимо обеспечить множественное связывание запроса для более чем одного DSA, DSA имеет возможность выбирать между различными стратегиями. Этот выбор всегда осуществляется в том случае, если DSA должен обработать более чем одну ссылку продолжения для одного **targetObject**. Подобная ситуация возникает при множественном связывании, вызванном декомпозицией NSSR при разрешении имени (как показано на рис. 22) или при декомпозиции запроса при оценке операции, выполняемой над многими объектами (см. рис. 23).

Задачей таких стратегий является найти решение для проблемы дублирования результатов и дублирования обработки при использовании теневой информации при множественном связывании запросов (вызванном либо NSSR, либо декомпозицией запроса). Например, как показано на рис. 22, DSA 1 проводит множественное связывание запроса к двум DSA с номерами 2 и 3 по той причине, что NSSR содержится в DSE B. Если разрешается использовать теневую информацию, то оба DSA (2 и 3) могут применять связанные операции для поддеревьев, начинающихся в X и Y.

Аналогично, на рис. 23 DSA 1 организует множественное связывание (в результате декомпозиции запроса) с двумя подчиненными ссылками, хранящимися в DSE X и Y. Если разрешается использование теневой информации, то оба DSA с номерами 2 и 3 могут применить связанные операции для двух поддеревьев, начинающихся в X и Y.

Для того чтобы решить проблему дублирования, DSA может выбрать одну из следующих стратегий при выполнении множественного связывания с запросами DSA, которые имеют один и тот же **targetObject**.

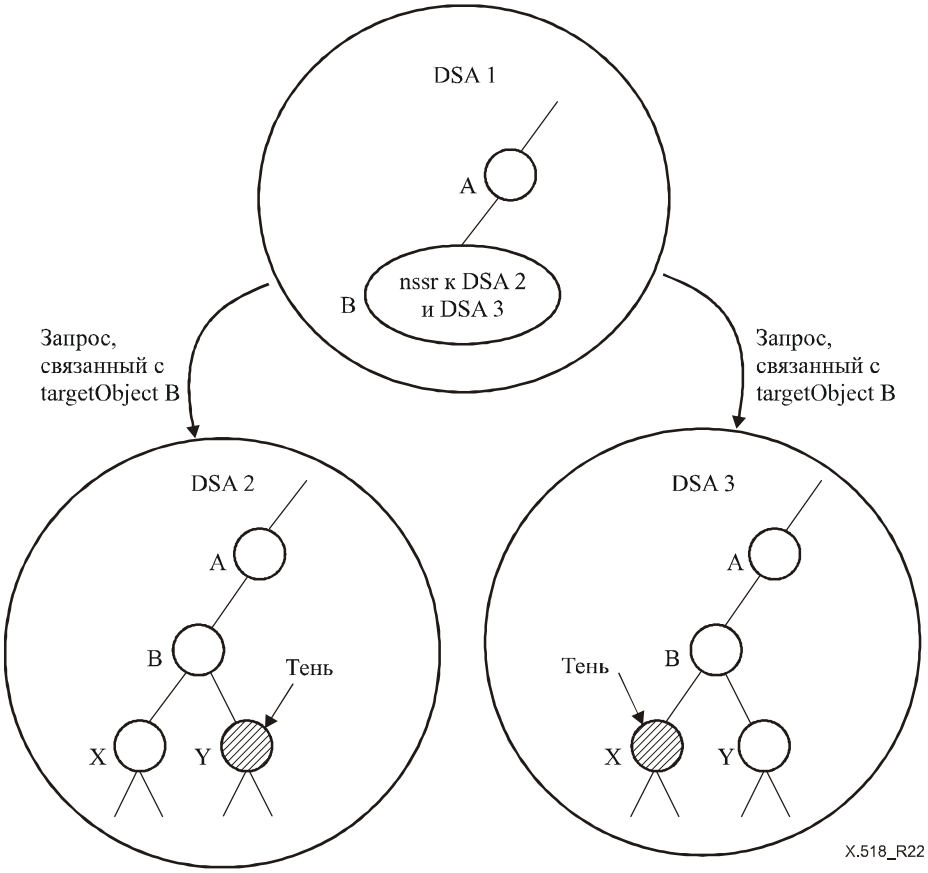


Рисунок 22 – Множественное связывание, вызванное NSSR при разрешении имени

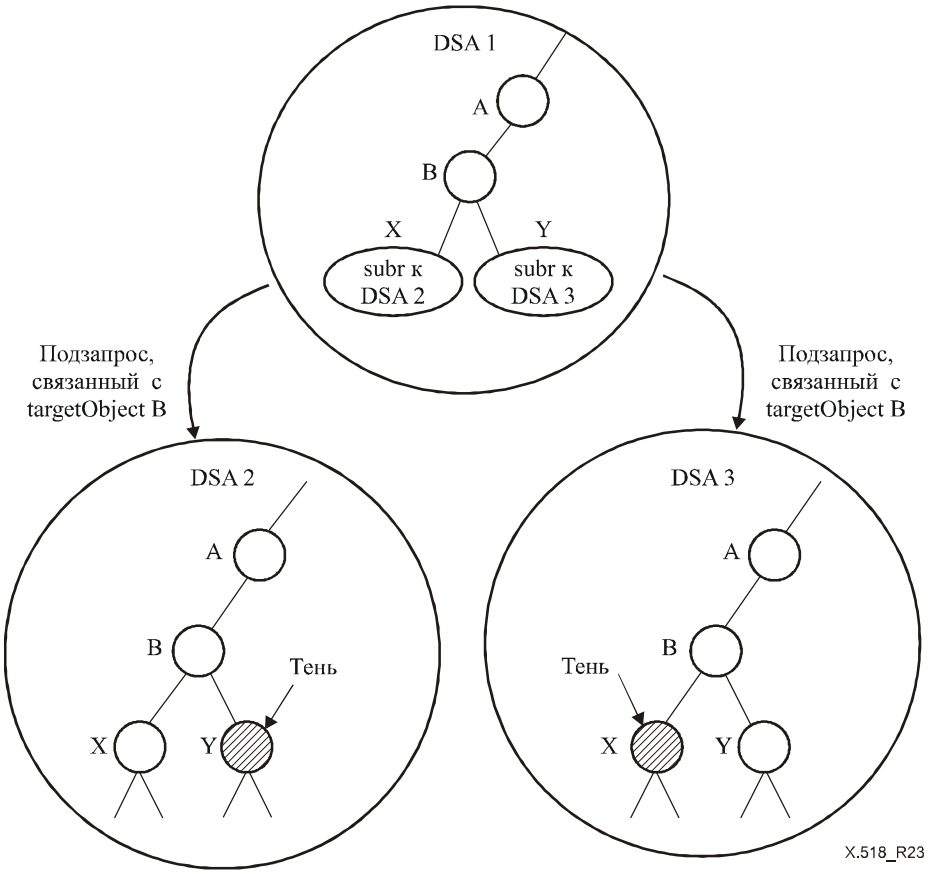


Рисунок 23 – Множественное связывание при декомпозиции запроса с использованием подчиненных ссылок

20.1.1 Стратегия Master only (только мастер)

DSA может выбрать данную стратегию для предотвращения использования теневой информации при проведении параллельного или последовательного связывания, вызванного декомпозицией NSSR или декомпозицией запроса при оценке Search (поиск) или List (список). Для данной стратегии во время оценки операции Search или List, компонент **excludeShadows** из **ChainingArguments** устанавливается равным **TRUE**. Если во время разрешения имени обнаруживаются NSSR, то DSA может установить **nameResolveOnMaster** равным **TRUE** с той целью, чтобы обеспечить перемещение по единственному пути. **nameResolveOnMaster** должен быть установлен как **TRUE** в том случае, если обнаруживаются NSSR и операция относится к операциям изменения Справочника. В любом случае, только тот (те) DSA, который содержит (первичную) главную статью (или статьи), которая имеет отношение к операции, должен выполнять операцию. Данная стратегия master only может использоваться как для параллельного, так и для последовательного множественного связывания.

ПРИМЕЧАНИЕ. – Устанавливая **nameResolveOnMaster** равным **TRUE**, мы устраняем возможность присутствия нескольких путей во время разрешения имени с помощью:

1) игнорирования теневых статей и копий статей, для которых разрешена операция записи; и
2) обеспечивая, что только один DSA может осуществлять разрешение имени в тех ситуациях, когда сложное распределение DIT разрешало бы проведение обработки для нескольких DSA.

Это обеспечивается за счет того, что позволяет продолжить разрешение имени тому DSA, который содержит (первичную) главную статью, которая соответствует первому **nextRDNToBeResolved** имени RDN для названия целевого объекта. Любой другой DSA не может проводить обработку – даже в том случае, если он содержит главные статьи, которые сопоставляются с большей частью названия целевого объекта.

20.1.2 Параллельная стратегия

Используя данную стратегию, DSA отправляет все связанные запросы с помощью параллельного множественного связывания. Данная стратегия может использоваться при оценке для операций Search (поиск) или List (список), и при разрешении имени для NSSR. Это позволяет использовать теневую информацию для обработки связанных запросов, однако может привести к дублированию действий и дублированию результатов операции. Если DSA выбирает данную стратегию, то при возврате результатов операции он должен удалить из них дублирующие результаты.

Если был запрошен подписанный результат, то удаление дублирующих результатов является невозможным. Когда при оценке операции Search (поиск) запрашиваются подписанные результаты, то DSA не должен выбирать данную стратегию, если только также не установлен **excludeShadows**.

20.1.3 Последовательная стратегия

Данная стратегия исключает возникновение дублирующих результатов с помощью использования последовательного множественного связывания при обработке связанных (под)запросов для декомпозиции Search (поиск) или для декомпозиции NSSR. Каждый связанный запрос обрабатывается последовательно, один за другим.

В случае декомпозиции NSSR, если в ответ на запрос возвращается результат или тяжелая ошибка, то для последующих запросов связывание не выполняется. Если возвращается мягкая ошибка, то в зависимости от локальной политики для следующего запроса может выполняться связывание или же инициатору запроса может быть возвращена мягкая ошибка.

В случае оценки для Search, компонент **exclusions** из **ChainingArguments** устанавливается как набор RDN, которые уже были обработаны. Это делается с помощью включения элементов из **ChainingResults.alreadySearched** в аргумент **exclusions** следующего связанного запроса. Это единственная стратегия, которая полностью исключает возможность дублирования при оценке Search (поиск).

Для оценки List (список) последовательная стратегия не определяется (хотя использование последовательного множественного связывания и допускается), так как вышестоящий DSA не имеет способа для исключения специфических подчиненных из результатов, возвращаемых последующими подзапросами List (список) (заметим, что **excludeShadows** не исключает специфические подчиненные, а использует более грубый способ, исключая все тени и копии, для которых разрешена операция записи).

20.2 Отправляем связанные подзапросы удаленному DSA

Когда DSA должен установить соединение с удаленным DSA, то перед отправкой подзапроса DSA должен выполнить операцию **dSABind**. Управление ассоциациями выходит за рамки данной спецификации Справочника. Ассоциация с другим DSA считается отсутствующей в том случае, если эта ассоциация не может быть установлена или же DSA по локальным причинам решает ее не устанавливать. В этом случае **dSABind** завершается неудачей. Решение относительно того, когда прекратить попытки устанавливать ассоциацию и объявить о том, что ассоциация отсутствует, относится к локальным решениям.

Когда DSA пытается применить **dSABind** к другому DSA и получает ошибку **directoryBindError**, то отправка подзапроса завершается неудачей.

20.3 Параметры процедуры

20.3.1 Аргументы

Данная процедура использует следующие аргументы:

- список ссылок продолжения для обработки в списке **NRcontinuationList** (для процедуры **Name Resolution Continuation Reference**) и в списке **SRcontinuationList** (соответственно для процедур **List Continuation Reference** и **Search Continuation Reference**);
- **CommonArguments** аргумента операции;

– аргументы **ChainingArguments**.

20.3.2 Результаты

Эти процедуры создают следующие результаты:

– список полученных результатов/ошибок для отправленных связанных запросов в том случае, если было выбрано связывание;
– обновленный список необработанных ссылок продолжения в списке **continuationList**.

20.3.3 Ошибки

Данные процедуры могут возвращать одну из следующих ошибок:

– ошибку **serviceError** с проблемой **outOfScope** в том случае, если был создан реферал, который не находится в пределах **scopeOfReferral**;
– ошибку службы **serviceError** с проблемой **ditError** в том случае, если была обнаружена недопустимая ссылка на знание;
– ошибку **nameError** с проблемой **noSuchObject** в том случае, если все подзапросы от декомпозиции NSSR возвратили **unableToProceed**;
– любую другую ошибку, которая возвращается связанным подзапросом;
– **referral** в том случае, если не было выбрано связывание и **operationProgress.nameResolutionPhase** установлен как **notStarted** или **proceeding**.

20.4 Определение процедур

Если **operationProgress.nameResolutionPhase** установлен как **notStarted** или **proceeding**, то следует действовать в соответствии с процедурой из п. 20.4.1 (процедура **Name Resolution Continuation Reference**). Операции обращения к множеству записей List и Search вызывают процедуры, определенные, соответственно, в пп. 20.4.2 и 20.4.3.

20.4.1 Процедура Name Resolution Continuation Reference (ссылка продолжения разрешения имени)

Процедура **Name Resolution Continuation Reference** состоит из этапов, показанных на рис. 24. Базовый принцип данной процедуры заключается в том, чтобы последовательно обработать набор ссылок продолжения, созданных при разрешении имени. Для каждой ссылки продолжения C, которая содержится в списке **NRcontinuationList**, необходимо последовательно выполнить показанные ниже этапы и этот процесс продолжается до тех пор, пока не будут обработаны все ссылки продолжения или же не будет возвращена ошибка или результат. Если были обработаны все ссылки, то возвращаемся в диспетчер операций **Operation Dispatcher**, для того чтобы продолжить процедуру **Result Merging** с целью обработки полученного результата или реферала.

1) Проверяем, установлено ли значение **chainingProhibited**. Если установлено, то DSA не разрешается проводить связывание. В соответствии с локальной стратегией **local policy**, возвращаем диспетчеру операций **Operation Dispatcher** либо ошибку **serviceError** с проблемой **chainingRequired**, либо **referral**.
2) Если значение **chainingProhibited** не установлено, то проверяем, разрешает ли связывание локальная стратегия **local policy**. Если связывание запрещено, то возвращаем **referral**. Если же локальная стратегия **local policy** разрешает связывание, то переходим к следующему этапу.
3) Обрабатываем каждую ссылку продолжения из списка ссылок продолжения, который хранится в списке **NRcontinuationList**. Если более не остается необработанных ссылок продолжения, то возвращаемся с ошибкой **serviceError**.
4) Обрабатываем следующую ссылку продолжения C из списка **NRcontinuationList**. Если она представляет NSSR, то переходим к этапу 5). Если она не является NSSR, то для ее обработки вызываем процедуру **APInfo**. Следует разделять следующие возможные варианты в зависимости от того, что возвращает процедура **APInfo**:
   – Если процедура **APInfo** возвращает **null result**, переходим к этапу 3) для обработки следующей ссылки продолжения.
   – Если процедура **APInfo** возвращает значение **error**, **referral** или **result**, то возвращаем это значение.
5) В этом случае ссылка продолжения относится к типу NSSR и DSA может выбирать между последовательным или параллельным связыванием в зависимости от локального выбора стратегии. Если NSSR должен обрабатываться последовательно, то переходим к этапу 6). Если обработка будет проводиться параллельно, то для каждой из **AccessPointInformation (API)** из NSSR вызывается процедура **APInfo**, таким образом осуществляется их параллельная обработка. Ожидаем завершения обработки всех API, т. е. ожидаем, пока все не возвратятся вызовы процедур **APInfo**. Для всех результатов, возвращенных вызванными процедурами **APInfo**, организуем проверку следующим образом:
   – Если все вызовы возвращают ошибку **serviceError** с проблемой **unableToProceed** и значение **partialNameResolution** равно **FALSE**, то возвращаем **nameError**.
   – Если все вызовы возвращают ошибку **serviceError** с проблемой **unableToProceed** и значение **partialNameResolution** равно **TRUE**, то в результате устанавливаем **partialName** равным **TRUE**, **nameResolutionPhase** равным **completed**, устанавливаем **entry suitable** (это применяется для **lastEntryFound**), и переходим к оценке соответствующей операции.

- Если получено один или более результатов **results**, то **discard possible duplicates** (отбрасываем возможные дубликаты) и возвращаем **result**.
- Если получена ошибка **error** и она не является ошибкой **serviceError** (например, **nameError**), то возвращаем **error**.
- В противном случае в соответствии с локальным выбором возвращаем диспетчеру операций **referral** или **serviceError**.

6) Выбираем следующий необработанный **API** из набора **API** данного NSSR и переходим к этапу 7). Если были обработаны все **API**, то проверяем, что все вызовы процедуры **APInfo** возвратили ошибку **serviceError** с проблемой **unableToProceed**.
   - Если это произошло и **partialNameResolution** равно **FALSE**, то запись не может быть найдена и возвращается ошибка **nameError**. Если это произошло и **partialnameResolution** равно **TRUE**, то в результате устанавливаем **partialName** равным **TRUE**, **nameResolutionPhase** равным **completed**, устанавливаем **entry suitable** (это применяется для **lastEntryFound**) и переходим к оценке соответствующей операции. Если это не произошло, то в соответствии с локальным выбором возвращаем **referral** или **serviceError**.
7) Вызываем процедуру **APInfo**. Следует различать между двумя возможными результатами вызова процедуры **APInfo**:
   - Если получена ошибка **serviceError** с проблемой **unableToProceed**, то необходимо попробовать другую точку доступа. Переходим к этапу 6).
   - Если получена ошибка **serviceError** с проблемой **busy**, **unavailable**, **unwillingToPerform** или **invalidReference**, то обозначенная проблема может носить переходный характер и попытка связывания запроса с другим DSA является предметом локального выбора. Если выбирается использование другого DSA, то переходим к этапу 6); в противном случае в зависимости от локального выбора возвращаем **referral** или **serviceError**.
   - Если получена ошибка, отличающаяся от ошибки **serviceError** с проблемой **busy**, **unavailable**, **unwillingToPerform**, **invalidReference** или **unableToProceed**, то данная ошибка должна возвращаться диспетчеру операций **Operation Dispatcher**. Если ошибка службы **serviceError** имеет значение **invalidReference**, то перед возвращением инициатору запроса она должна быть преобразована в ошибку **ditError**.
   - Если получен результат **result** или реферал **referral**, то возвращаем их диспетчеру операций **Operation Dispatcher**.

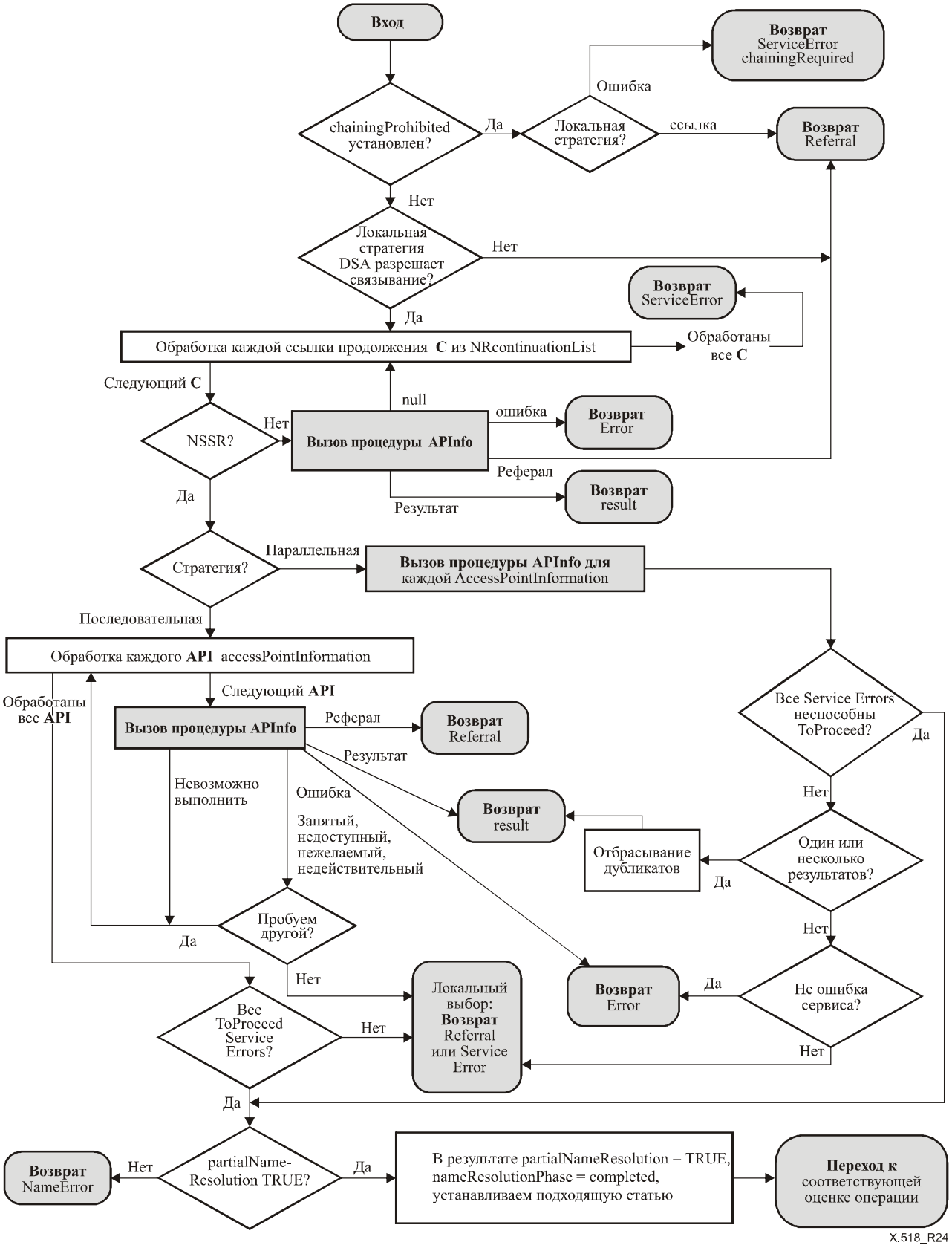


Рисунок 24 – Процедура Name Resolution Continuation Reference

20.4.2 Процедура List Continuation Reference

Процедура **List Continuation Reference** состоит из этапов, показанных на рис. 25. Эта процедура вызывается в том случае, если запрос List не может быть выполнен в локальной DSA и к списку **SRcontinuationList** был добавлен

набор ссылок продолжения для связывания или реферал. Все эти ссылки продолжения (CR) имеют один и тот же **targetObject**. Если CR относится к **referenceType nssr**, то она имеет одно или более значений **AccessPointInformation** (**API**), тогда как другие типы CR содержат только одну **API**. Каждая из этих **API** извлекается и рассматривается на предмет связывания или реферала.

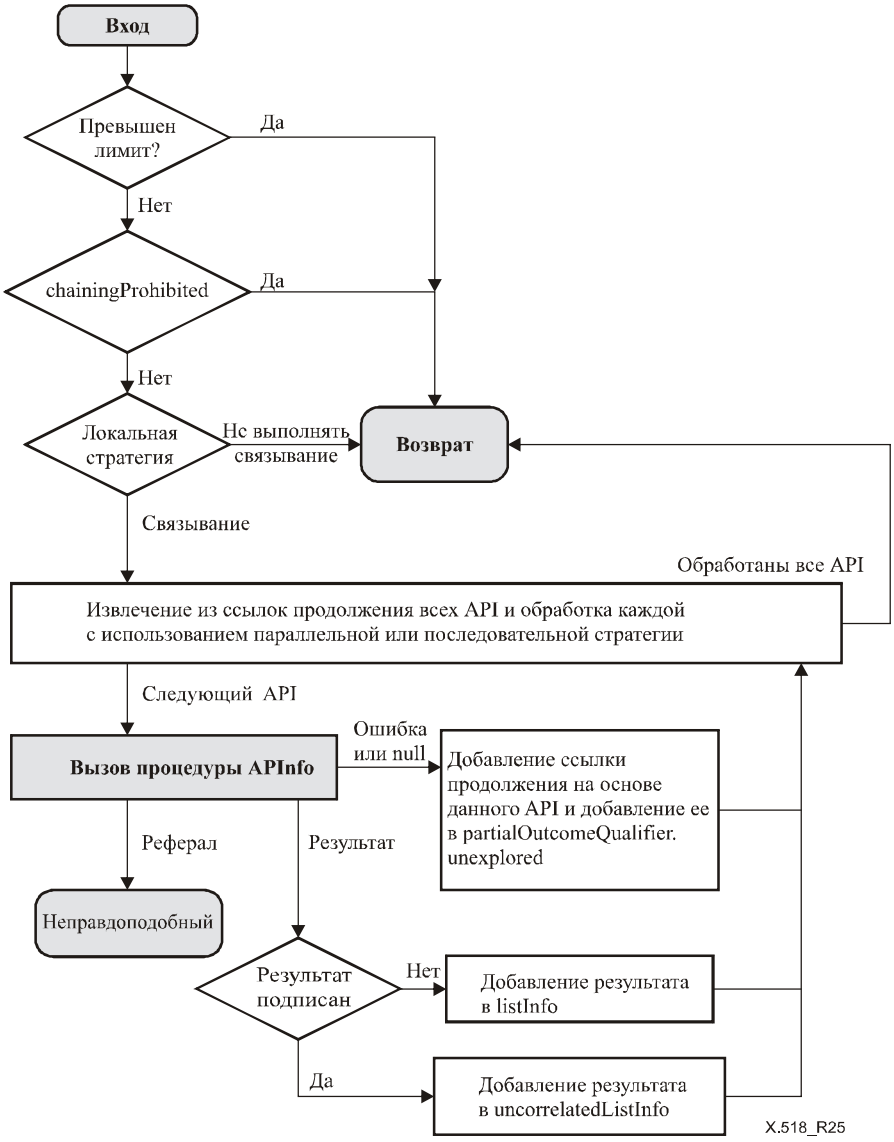


Рисунок 25 – Процедура List Continuation Reference

Должны быть выполнены следующие этапы:

1) Если был превышен какой-то из установленных лимитов, то возвращаемся в диспетчер операций **Operation Dispatcher** и продолжаем процедуру **Result Merging**.
2) Если установлен флаг **chainingProhibited**, расположенный в **CommonArguments.serviceControls**, или же DSA решает не выполнять связывание, основываясь на своей локальной операционной стратегии, тогда DSA должен непосредственно возвратиться к диспетчеру операций **Operation Dispatcher** и продолжить выполнение процедуры **Result Merging**.
3) Создаем набор значений **AccessPointInformation** с помощью компонента **accessPoints** для каждой ссылки продолжения, которая содержится в списке **SRcontinuationList**.

   Для обработки каждого **API** используем параллельную или последовательную стратегию следующим образом:

   i) Вызываем процедуру **APInfo** для следующего в наборе **API**.

   ii) Если возвращается результат и если он не снабжен подписью, то добавляем его к **listInfo**, если же результат подписан, то добавляем его к **uncorrelatedListInfo**.

   iii) Если возвращается ошибка или ноль, то это означает, что **APInfo** уже пыталась использовать все точки доступа из **API** и не достигла успеха. Основываясь на локальной функциональной стратегии и стратегии безопасности, можно либо игнорировать это, либо перейти к следующему **API**, или же добавить ссылку продолжения, основанную на данной **API**, к **partialOutcomeQualifier**.

ПРИМЕЧАНИЕ. – Невозможно получить обратно реферал от **APInfo**. Любой "реферал" должен передаваться в форме **unexplored** из **partialOutcomeQualifier**.

4) Если обработаны все **API**, то возвращаемся в диспетчер операций **Operation Dispatcher**.

20.4.3 Процедура Search Continuation Reference (поиск ссылки продолжения)

Процедура **Search Continuation Reference** состоит из этапов, показанных на рис. 26. Данная процедура вызывается тогда, когда запрос Search не может быть выполнен в локальном DSA и набор ссылок продолжения был добавлен к списку **SRcontinuationList** с целью связывания данного запроса или для реферала. Данная процедура подобна процедуре **List Continuation Reference**. Отличие состоит в том, что в данном случае ссылки продолжения из списка **SRcontinuationList** могут иметь различные значения **targetObject**. Таким образом, ссылки продолжения сортируются в наборы ссылок продолжения с одинаковым **targetObject**. Кроме этого, определяется использование **exclusions** в аргументах связывания и **alreadySearched** в результатах связывания, так как это является важной стратегией при поиске. Использование **exclusions** и **alreadySearched** применяется при обработке каждого набора ссылок продолжения с одинаковым **targetObject**.

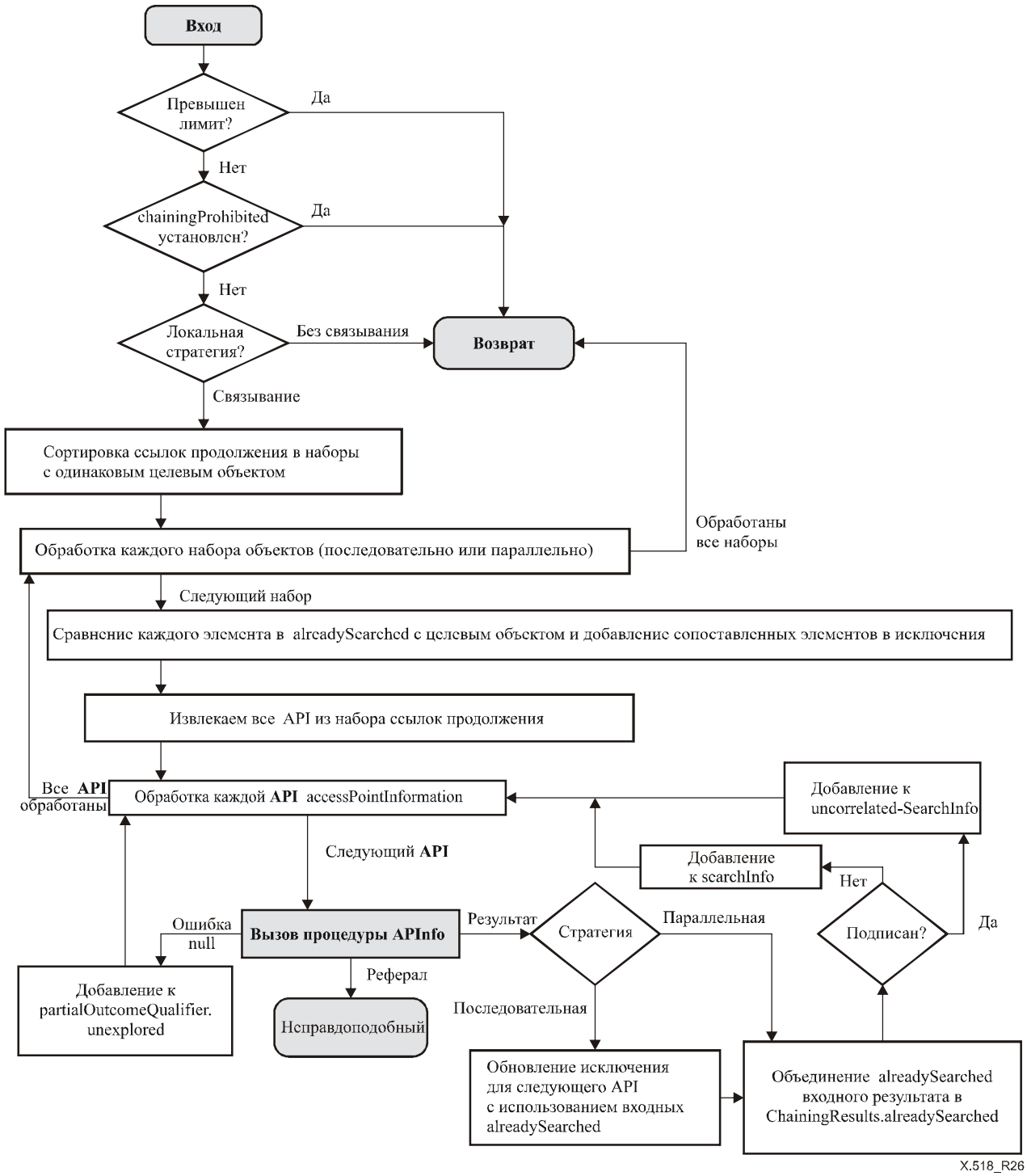


Рисунок 26 – Процедура Search Continuation Reference

Должны быть выполнены следующие этапы:

1) Если был превышен какой-то из установленных лимитов, то возвращаемся в диспетчер операций **Operation Dispatcher** для продолжения процедуры **Result Merging**.
2) Если установлен флаг **chainingProhibited** в **CommonArguments.serviceControls** или DSA решает не осуществлять связывание по причине своей локальной операционной стратегии, то DSA должен возвратиться непосредственно в диспетчер операций **Operation Dispatcher** для продолжения процедуры **Result Merging**.
3) Выполняем сортировку ссылок продолжения в списке **SRcontinuationList** с целью получить наборы ссылок продолжения, в которых все ссылки продолжения имеют одинаковый **targetObject**. ссылки продолжения типа **ditBridge** не включаются в такие наборы, однако каждая из таких ссылок продолжения образует собственный набор. В каждом из таких наборов удаляем все дублирующие ссылки.

   ПРИМЕЧАНИЕ 1. – Если одно или более из значений **targetObject** не являются первичным RDN, тогда такая сортировка может не являться точной. Сортировка должна также учитывать альтернативные выделенные имена RDN, если только они известны.
4) Для каждого набора ссылок продолжения создаем набор значений **AccessPointInformation** на основании компонента **accessPoints** каждой ссылки продолжения в данном наборе, и выбираем для последующей обработки последовательную или параллельную стратегию. Если выбирается параллельная стратегия, то пропускаем описанные ниже этапы, которые применимы только для последовательной стратегии.
   a) Если выбирается последовательная стратегия, то поддерживаем локальную переменную **localExclusions** для каждого набора ссылок продолжения, имеющих одинаковое значение **targetObject**. Первоначально **localExclusions** устанавливается как **exclusions** из входящего связанного запроса (если он существует), и как все поддеревья с локальным поиском, расположенные непосредственно ниже **targetObject**.
   b) Если используется последовательная стратегия, то сравниваем **targetObject** со всеми элементами **localExclusions** и удаляем те элементы, которые не содержат в качестве префикса **targetObject**. Это релевантные исключения для текущего целевого объекта.
   c) Извлекаем все **API** из ссылок продолжения для набора с текущим целевым объектом.
   d) Осуществляем цикл для каждого **API**. В нем для каждого **API**:
      i) Вызываем **APInfo**.
      ii) Если возвращается результат, то если он не снабжен подписью, добавляем результат к **searchInfo**, а если результат подписан, то добавляем его к **uncorrelatedSearchInfo**. В том случае, если выбрана последовательная стратегия, то обновляем **localExclusions** с использованием **alreadySearched** из входящего ответа, а также объединяем **alreadySearched** из входящего ответа с **ChainingResults.alreadySearched** для данного DSA. После этого переходим к следующему **API**.
      iii) Если возвращается ошибка или ноль, то это означает, что **APInfo** уже использовал все точки доступа в API и не достиг успеха. Основываясь на локальной функциональной стратегии и стратегии безопасности, можно либо проигнорировать это и перейти к следующему **API**, либо же на основании данного **API** добавить ссылку продолжения к **partialOutcomeQualifier**.

         ПРИМЕЧАНИЕ 2. – Невозможно получить реферал обратно от **APInfo**. Любой "реферал" должен поступать в виде **unexplored** в **partialOutcomeQualifier**.
   e) После того, как были обработаны все **API**, переходим к следующему набору ссылок продолжения с одинаковым значением **targetObject**.
5) Когда обработаны все ссылки продолжения, то возвращаемся к диспетчеру операций **Operation Dispatcher**.

20.4.4 Процедура APInfo

Данная процедура вызывается для обработки **AccessPointInformation**, которая содержит одну или более точек доступа (см. рис. 27). Они обрабатываются одна за другой до тех пор, пока не будет возвращен результат или ошибка. Если ошибка является ошибкой службы, то попытка использовать другую точку доступа может привести к успеху, и в этом случае проверяются дополнительные точки доступа до тех пор, пока это разрешает локальная операционная стратегия:

1) Выполняем проверку петли. Если обнаруживается петля, то возвращаем **serviceError** с проблемой **loopDetected**. В противном случае переходим к этапу 2).
2) Обрабатываем каждую из точек доступа на основании информации о точке доступа. Если все точки доступа уже обработаны, то возвращаем **null result**. Если остаются точки доступа для обработки, то переходим к этапу 3).
3) Проверяем, разрешает ли локальная стратегия связывание с данной точкой доступа. Такая проверка должна учитывать настройки управления службой и аргументов связывания (например, **chainingProhibited**, **preferChaining**, находится ли точка доступа в пределах **localScope** или нет, **excludeShadows**). Если локальная стратегия или соответствующие настройки управления службой не позволяют использовать данную точку доступа, то игнорируем эту точку доступа и переходим к этапу 2). Если же использование данной точки доступа допускается, то переходим к этапу 4).
4) Если локальная стратегия выбирает стратегию "master only", то устанавливаем аргумент связывания **excludeShadows** равным **TRUE**.

Если значение **nameResolutionPhase** не равно **completed** и стратегия заключается в том, чтобы продолжать разрешение имени для главных статей, то устанавливаем **nameResolveOnMaster** равным **TRUE**.

Аргумент связывания **nameResolveOnMaster** должен быть установлен как **TRUE**, если выполняется одно из следующих условий:

– во входящем аргументе связывания значение **nameResolutionPhase** равно **proceeding** и **nameResolveOnMaster** равен **TRUE**; или

– операция является одной из операций модификации, значение **referenceType** для отправляемого запроса равно NSSR, и при этом используется параллельная стратегия.

ПРИМЕЧАНИЕ. – Данный метод использования **nameResolveOnMaster** направлен на запрещение многократного применения операций изменения из-за наличия NSSR.

5) Создаем связанный запрос и пытаемся его отправить:

a) Выполняем процедуру предотвращения образования петли с помощью проверки того, существует ли в полученном **ChainingArguments** в **traceInformation** элемент с аналогичными значениями для **targetObject** и **operationProgress**. Если результирующий запрос [(как это описывается в этапе 5), c)] приводит к образованию петли, то DSA должен либо возвратить ошибку **serviceError** с проблемой **loopDetected** тому DUA/клиенту LDAP/DSA, который выступал в качестве инициатора запроса, или же игнорировать данную точку доступа и перейти к этапу 2).

b) Если связываемый запрос или подзапрос является результатом выполнения реферала, то понадобится еще одна проверка для того, чтобы избежать образования петли. Проверяем, имеется ли в **referralRequests** элемент с аналогичными значениями **targetObject**, **operationProgress** и целевого DSA. Если это так, то предпринимаем действия, описанные в пункте a). Если нет, то добавляем новый **TraceItem** к **referralRequests** со следующими компонентами:

– **targetObject** и **operationProgress** установлены как значения в связанном запросе/подзапросе;

– **dsa** устанавливается как имя DSA, с которым будет связываться запрос/подзапрос.

c) После успешного окончания Bind (привязывания), DSA должен выдать связанную операцию с тем же типом операции, которая обрабатывается в данный момент, при этом устанавливаются следующие параметры:

– аргумент операции в связанной операции устанавливается как аргумент в полученной операции;

– **ChainingArguments.originator** устанавливается таким же, как у полученного;

– **ChainingArguments.targetObject** устанавливается как **targetObject** ссылки продолжения;

– **ChainingArguments.operationProgress** устанавливается равным значению **operationProgress** в ссылке продолжения;

– **ChainingArguments.traceInformation** устанавливается равным трассировочной информации с учетом обновления процедурой **Request Validation** в том случае, если ссылка продолжения не относится к типу **ditBridge**, в противном случае данный компонент отсутствует;

– **ChainingArguments.aliasDereferenced** устанавливается равным обновленному значению **aliasDereferenced**, которое обновляется локальным образом;

– **ChainingArguments.returnCrossRefs** устанавливается в соответствии с локальным выбором;

– **ChainingArguments.referenceType** устанавливается равным **referenceType** ссылки продолжения;

– **ChainingArguments.timeLimit** устанавливается равным значению полученного лимита времени **timeLimit**;

– **SecurityParameters** устанавливается как значение полученных **SecurityParameters**.

6) Если запрос не может быть выдан успешно, то переходим к этапу 7). Если запрос может быть успешно выдан, то переходим к этапу 8).

7) Продолжать или нет является локальным вопросом. Если DSA решает продолжать, то ошибка игнорируется и попытка выполняется для следующей точки доступа. Переходим к этапу 2). Если DSA решает не выполнять попытку для следующей точки доступа, то в зависимости от локальной стратегии решаем, возвращать ли соответствующий **referral** или **serviceError** тому, кто вызвал данную процедуру.

8) Если запрос не может быть отправлен успешно, то DSA ожидает ответа и обрабатывает его:

a) Если получен результат **result**, то **result** возвращается тому, кто вызвал данную процедуру.

b) Если получена ошибка **serviceError** с проблемой **busy**, **unavailable**, **unwillingToPerform** или **invalidReference**, то переходим к этапу 7).

c) Если получен **referral** и **returnToDUA** установлен как **TRUE**, то принимающий DSA не должен выполнять никаких действий над рефералом, а должен возвратить реферал тому, кто инициировал запрос.

d) Если был получен **referral** и **returnToDUA** установлен как **FALSE**, то применяются те же соображения локальной стратегии, которые использовались на этапе 3) (принимаются во внимание управление службой, аргументы связывания, стратегия связывания и так далее). Если принимается решение не разыменовывать **referral**, то возвращаем **referral** вызывающей стороне. Если принимается решение разыменовать **referral**, то очищаем список **NRcontinuationList**, помещаем полученную ссылку продолжения в реферал, содержащийся в списке **NRcontinuationList** и вызываем процедуру **Name Resolution Continuation Reference**. Это может привести к получению **result**, **referral**, **serviceError** или же других ошибок **error**. Вне зависимости от того, что именно было получено при вызове процедуры **Name Resolution Continuation Reference**, оно должно быть возвращено вызывающей стороне.

e) Если возникает какая-то другая ошибка **error**, она должна быть возвращена вызывающей стороне.

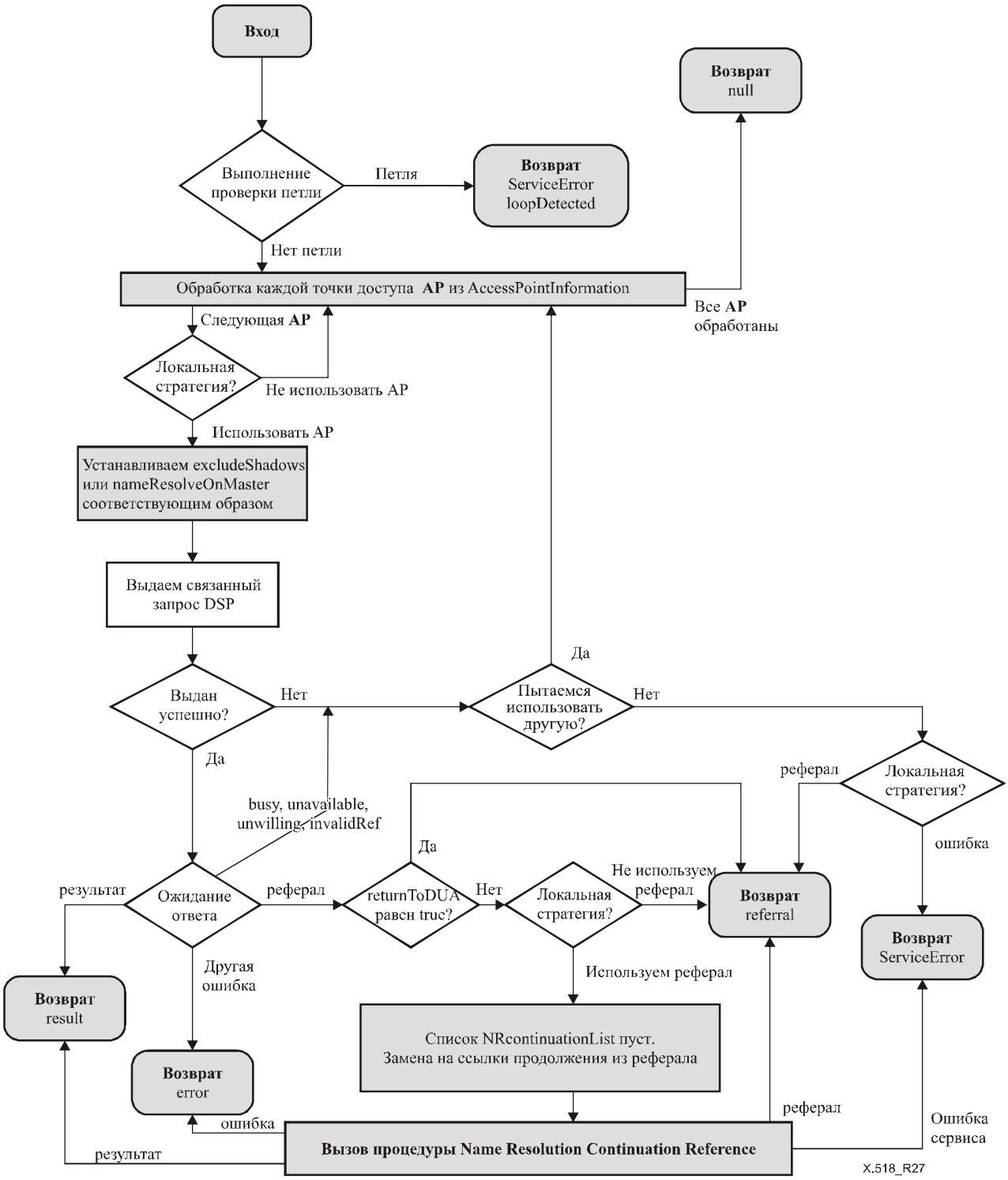


Рисунок 27 – Процедура APInfo

20.5 Процедура Abandon (прекратить)

Данная процедура вызывается в том случае, когда был получен запрос прекращения. Процедура состоит из следующих этапов, показанных на рис. 28:

1) При получении запроса **abandon**, который ссылается на неизвестную операцию, запрашивающей стороне должна быть возвращена ошибка **abandonFailed** с проблемой **noSuchOperation**.

2) Если на запрос о прекращении уже был дан ответ, и DSA располагает информацией об этом, то запрашивающей стороне может быть возвращена ошибка **abandonError** с проблемой **tooLate**.

3) Если запрос Abandon (прекратить) не является действительным, т. е. прекращение направлено запросу, который не является запросом обращения, то запрашивающей стороне должна быть возвращена ошибка **abandonFailed** с проблемой **cannotAbandon**.

4) Если у DSA имеются текущие связанные (под)запросы в тот момент, когда он получает действительный запрос прекращения для первоначального запроса, и DSA принимает решение попробовать выполнить прекращение, то он может отправить запросы прекращения ни одному, некоторым или всем текущим (под)запросам для рассматриваемой операции, и затем ожидать ответа на запрос прекращения и на текущие (под)запросы. В любой момент в течение этой операции DSA может отправить запрашивающей стороне результат прекращения и ошибку **abandonFailed**, а затем отбрасывать ответы на отправленные запросы прекращения и на текущие (под)запросы по мере того, как они поступают.

   Если DSA решает не посылать ответы запрашивающей стороне до тех пор, пока не останется текущих (под)запросов, то в качестве опции он может отправить запрашивающей стороне ошибку **abandonedFailed** в том случае, если на все отправленные запросы **abandon** были получены ответы с ошибкой **abandonedFailed** и если не выполнялось никаких локальных операций прекращения.

   Если запрашивающей стороне возвращается ошибка **AbandonedFailed**, то первоначальный запрос должен рассматриваться таким образом, как будто бы запрос прекращения никогда не был получен.

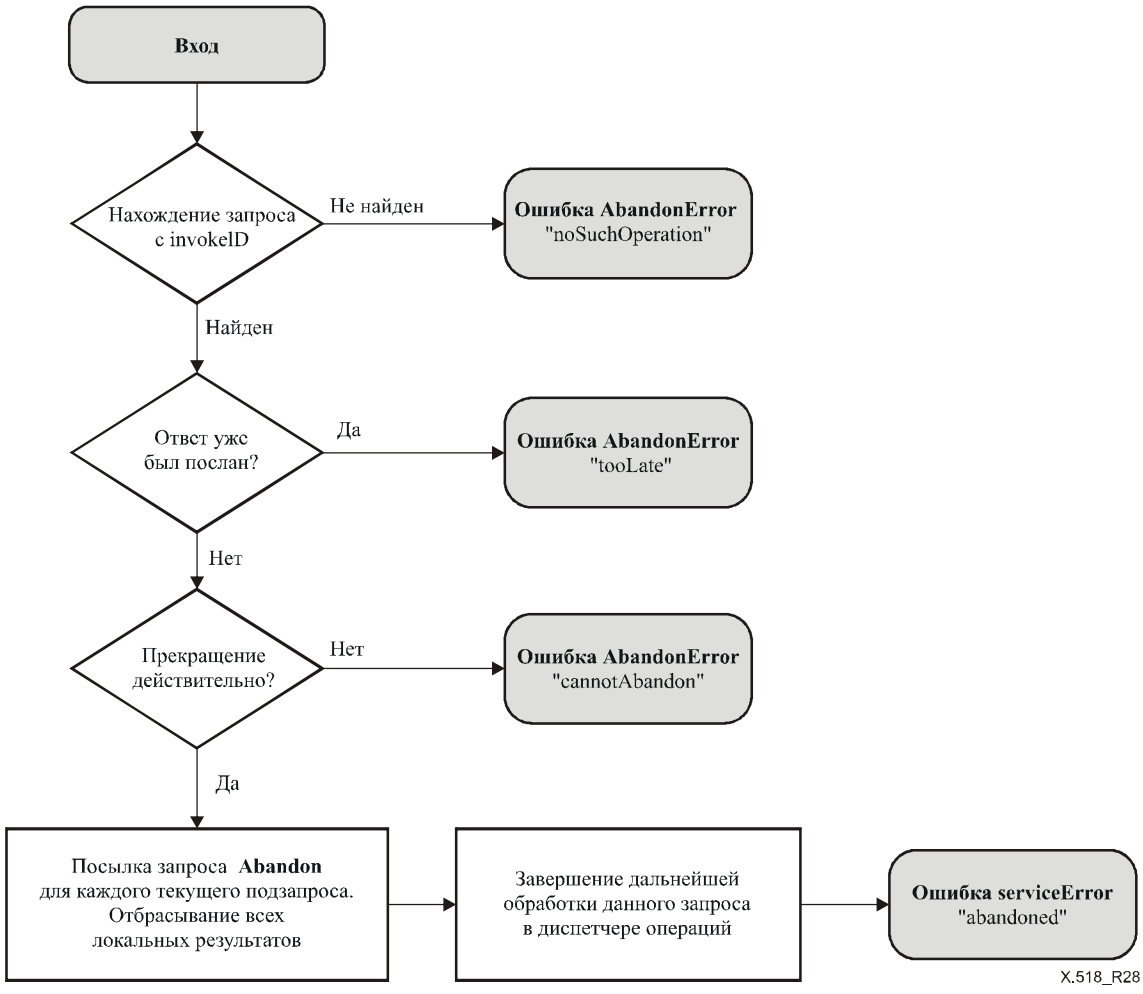


Рисунок 28 – Процедура Abandon

21 Процедура Results Merging (объединение результатов)

Процедура **Result Merging** (объединение результатов) показана на рис. 29 и вызывается вслед за одной из процедур **Continuation Reference**. Данная процедура удаляет дублирующие результаты если они не подписаны и если в **partialOutcomeQualifier.unexplored** существуют дополнительные ссылки продолжения. В этом случае вызывается(ются) соответствующая(ие) процедура(ы) ссылки продолжения, если только локальная операционная стратегия разрешает следующее:

1) Если операция является операцией List (список), то переходим к этапу 2); если операция является операцией Search (поиск), то переходим к этапу 3); в противном случае возвращаем результат, который был передан как входной параметр для процедуры **Result Merging**.

2) Операция является операцией List (список). Удаляем все дубликаты, отдавая предпочтение главной информации перед теневой информацией.

   Если результат операции был сгенерирован локально и если он содержит ссылки продолжения, то они не будет использоваться для связывания, а просто возвращаются пользователю. В этом случае переходим к этапу 6).

   Если результат операции был получен как результат операции Chained List (связанный список), то результат может содержать ссылки продолжения. В этом случае проверяем, установлено ли управление службой **preferChaining**. Если оно равно **TRUE**, то ссылки продолжения должны использоваться DSA для связывания. Переходим к этапу 4).

3) Операция является операцией Search (поиск). Удаляем все дубликаты, отдавая предпочтение главной информации над теневой информацией. Если существуют проблемы с лимитами, тогда возвращаем результат. В противном случае переходим к этапу 4).

4) Обрабатываем каждую ссылку продолжения, которая находится в **partialOutcomeQualifier.unexplored** результата любой связанной операции. Если локальная стратегия принимает решение не использовать эту ссылку для связывания, то игнорируем и переходим к другой ссылке продолжения. Если же локальная политика разрешает использование ссылок продолжения для связывания, то выполняем следующее:

   Проверяем значение **nameResolutionPhase**, которое передается в ссылке продолжения. Если оно не равно **notStarted** или **proceeding**, то добавляем его к списку ссылок продолжения, который передается процедуре **Name Resolution Continuation** (список **NRcontinuationList**). Если значение **nameResolutionPhase** равно **completed**, тогда добавляем ссылку продолжения к списку ссылок продолжения, который передается в процедуру продолжения подзапроса (**SRcontinuationList**).

   Продолжаем до тех пор, пока не будут обработаны все ссылки продолжения.

5) Если в списке **SrcontinuationList** существуют ссылки продолжения для обработки, то проверяем тип операции. Если операция относится к операциям List (список), то вызываем процедуру **List Continuation Reference** и переходим к этапу 2). Если операция является операцией Search (поиск), то вызываем процедуру **Search Continuation Reference** и переходим к этапу 3).

   Если список **SRcontinuationList** пуст, то проверяем, имеются ли ссылки продолжения в списке **NRcontinuationList**. Если да, то вызываем процедуру **Name Resolution Continuation Reference** и переходим к этапу 3).

   Если оба списка продолжения пусты, то переходим к этапу 6).

6) Проверяем, является ли результат пустым. Если он не пустой, то возвращаем данный результат. Если же он пуст, то либо возвращаем **null result** в том случае, если это позволяют контроль доступа и локальная стратегия, или же возвращаем соответствующую ошибку.

В том случае, если DSA получает результаты поиска или списка от других DSA и эти результаты имеют параметры, неизвестные для данного DSA, то должны возвращаться некоррелированные результаты. Иначе DSA должен выполнить объединение в том случае, если результаты поиска не подписаны или же если DSA является начальным исполнителем, которому разрешается удалять подписи (см. п. 7.9 Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3).

Если DSA получило неподписанные и некоррелированные результаты от другого DSA, который не способен выполнить их консолидацию, то он должен выполнить слияние в том случае, если он обладает надлежащим знанием о всех параметрах некоррелированных результатов.

Если DSA получает от других DSA неподписанные результаты и также возможно имеет локальный результат, то при генерации количества записей, которые должны возвращаться как значение **entryCount** в сгенерированном DSA **PartialOutcomeQualifier**, DSA должен использовать сумму всех полученных значений **entryCount**, локального результата и количества записей полученных от DSA, которые не возвратили значение **entryCount**, а затем компенсировать наличие дублирующих статей. Если DSA является начальным исполнителем, и были запрошены постраничные результаты, то он должен включить количество записей для подписанных результатов от других DSA.

Если запрашивались постраничные результаты и ни один DSA не встретил проблем с лимитами, то DSA должен выбрать опцию **exact** для параметра **entryCount**. Эта величина должна быть присвоена каждой возвращенной странице.

Если же один или более DSA столкнулся с проблемами лимитов, тогда:

– если все DSA, которые столкнулись с проблемой лимитов, возвратили **entryCount** со значениями, равными **exact** или **bestEstimate**, то следует выбрать **bestEstimate** в том случае, если этот выбор был сделан хотя бы одним DSA, в противном случае выбирается опция **exact**;

– если только один DSA из тех, которые столкнулись с проблемой лимитов, возвратил при этом **entryCount** с опцией **lowEstimate** или же не возвратил **entryCount**, то выбирается опция **lowEstimate**.

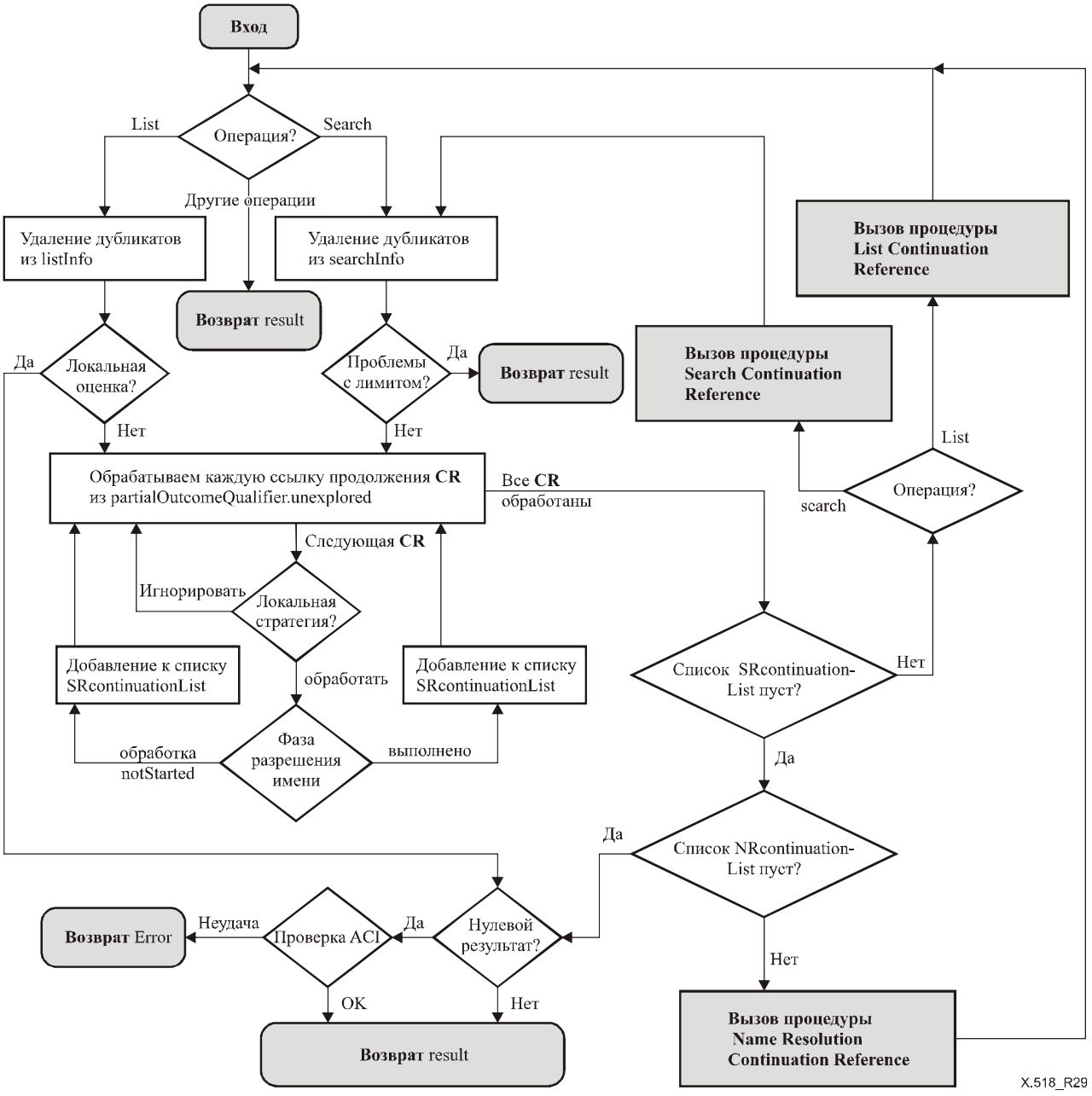


Рисунок 29 – Процедура Results Merging

22 Процедуры распределенной аутентификации

В данном разделе описываются процедуры, необходимые для поддержки служб распределенной аутентификации Справочника. Эти службы и, следовательно, их процедуры, разбиваются на следующие категории:

– аутентификация инициатора, которая поддерживается в незащищенной форме (основанная на простой идентификации) или защищенной (основанная на цифровой подписи) форме, и

– аутентификация результатов, которые защищаются аналогичным образом (опять на основе цифровых подписей).

22.1 Аутентификация инициатора

22.1.1 Аутентификация на основе идентификации

Службы аутентификации на основе идентификации позволяют DSA аутентифицировать того, кто являлся первоначальным инициатором запроса информации, с целью осуществления локального контроля доступа. DSA, который хочет использовать такую службу, должен принять следующие процедуры:

– Для DSA, который требует аутентификации DAP или запроса LDAP, DSA получает выделенное имя инициатора запроса с помощью процедур Bind (привязать) в то время, когда устанавливается ассоциация DUA (DUA к DSA) или ассоциация клиента LDAP (клиент LDAP к DSA). Успешное завершение этих процедур ни в коей степени не причиняет ущерб уровню аутентификации, который затем может потребоваться для обработки операций с использованием данной ассоциации.

– DSA, у которого существует ассоциация с DUA или клиентом LDAP, должен вставить выделенное имя инициатора запроса в поле инициатора в **ChainingArguments** для всех последующих связанных операций с другими DSA.

– При получении связанной операции DSA может выполнить эту операцию или же нет, в зависимости от определения прав доступа (механизм, определенный локально). Если результат является удовлетворительным, то может возвращаться ошибка **securityError** с проблемой **insufficientAccessRights**.

22.1.2 Аутентификация инициатора на основе подписи

Данная служба аутентификации инициатора на основе подписи позволяет DSA аутентифицировать (безопасным способом) инициатора определенного запроса о службе. Процедуры, которые должны осуществляться DSA при выполнении этой службы описываются в данном разделе.

Служба аутентификации на основе подписи вызывается DUA с использованием варианта **PROTECTED** для запроса к службе с возможностью защиты с установленными значениями **DirQOP signed** или **signedAndEncrypted**.

DSA при получении подписанного запроса от другого DSA перед выполнением операции должен удалить подпись этого DSA. Допуская, что любая проверка подписи приносит удовлетворительный результат, DSA продолжит выполнение операции. Если при этом DSA необходимо осуществить связывание, то для каждой связанной операции набор аргументов конструируется следующим образом:

– DSA создает набор аргументов, которые могут быть при необходимости подписаны; этот набор аргументов состоит из набора входных подписанных аргументов вместе с модифицированными **ChainingArguments**.

В том случае, если DSA способен добавлять информацию к ответу, то для определения прав доступа к этой информации может использоваться аутентификация инициатора, основанная на подписанном запросе службы.

Если DSA получает неподписанный запрос службы, который будет отправлен только в зависимости от аутентификации инициатора, то должна возвращаться ошибка **securityError** с проблемой **protectionRequired**.

22.2 Аутентификация результатов

Эта служба предоставляется для того, чтобы обеспечить заказчикам операций Справочника (DUA, клиент LDAP или DSA) возможность проверить (безопасным способом на основе технологий цифровой подписи) источник результатов. Служба аутентификации результатов может запрашиваться вне зависимости от того, используется ли аутентификация инициатора.

Служба аутентификации результатов инициируется с использованием подписанного значения компонента **protectionRequest**, который содержится в наборе аргументов операций Справочника; DSA, который получает операцию с такой установленной опцией, может по своему выбору подписывать все последующие результаты. Опция подписи в защищенном запросе служит в качестве индикатора, который сообщает DSA о предпочтениях заказчика, сам DSA может действительно подписывать или не подписывать все последующие результаты.

В том случае, если DSA выполняет связывание, то DSA имеет набор вариантов для выбора формы, в которой результаты отсылаются обратно заказчику, а именно:

a) заказчику возвращается составной ответ (подписанный или неподписанный);

b) заказчику возвращается набор из двух или более неупорядоченных частичных ответов (подписанных или не подписанных); в пределах этого набора ноль или более членов могут являться подписанными, а ноль или один является неподписанным. В том случае, если присутствует неподписанный частичный результат, то этот член может являться упорядочением одного или нескольких неподписанных частичных ответов, которые были получены от других DSA, или же представлены этим DSA, или же возможна комбинация двух этих вариантов.

В том случае, когда DSA выполняет объединение связанных записей, то выполняющий объединение DSA может подписать результат.

РАЗДЕЛ 6 – АДМИНИСТРИРОВАНИЕ ЗНАНИЯМИ

23 Обзор администрирования знанием

Для того чтобы работать с широко распределенным Справочником, обеспечивая при этом в достаточной степени согласованность и производительность, необходимы процедуры, которые обеспечивают создание, поддержание и расширение знания, которое хранится в каждом DSA. Для администрирования знания DSA совместно используются следующие механизмы.

a) *Иерархические и неспецифические иерархические операционные связывания* – Эти процедуры и протоколы определяются в пп. 24 и 25. Они используются для создания и поддержания подчиненных ссылок, неспецифических подчиненных ссылок и непосредственных вышестоящих ссылок, а также в качестве информации о контекстных префиксах для контекстов именования. Такие операционные связывания устанавливаются между главным DSA, хранящем контексты именования, которые иерархически связаны друг с другом в качестве непосредственно подчиненных непосредственного вышестоящего объекта. Процедуры могут запускаться в качестве стороннего эффекта при изменении RDN или при удалении или добавлении статьи, непосредственно вышестоящий объект для которой не содержится в том же DSA, в котором хранится данная статья.

b) *Теневые операционные связывания* – Эти процедуры и протоколы определяются в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9. Они используются для создания и поддержания ссылок на знания двумя различными способами. В первом случае, в качестве побочного эффекта от установления (или завершения) теневых соглашений, добавляются (или удаляются) точки доступа из операционных атрибутов **consumerKnowledge** и (в качестве опции) **secondaryShadow**. Эта информация может затем использоваться рассмотренными выше процедурами и протоколами для обновления подчиненной ссылки в вышестоящем главном DSA и непосредственно вышестоящей ссылки в подчиненном главном DSA. Во втором случае, DISP распространяет ссылку на знания, которая хранится главным DSA, к теневым потребительским DSA.

c) *Перекрестные ссылки* – Распределение перекрестных ссылок является функцией DSP, его использование для создания и поддержания перекрестных ссылок суммируется в п. 23.2.

ПРИМЕЧАНИЕ. – Механизмы для инициализации и поддержания вышестоящей ссылки и **myAccessPoint** не относятся к предмету данной спецификации Справочника.

23.1 Поддержание ссылок на знания

В данном подразделе мы описываем, как DOP используется для поддержания операционных атрибутов DSA, которые используются для выражения знания. Простой пример отношения между атрибутами знания и протоколами, которые используются для их поддержки, приводится в Приложении E.

23.1.1 Поддержание знания потребителя поставщиком и главными DSA

Ссылка потребителя выражается с помощью значения атрибута **consumerKnowledge**, который хранится DSA теневого поставщика и который связан с контекстным префиксом для контекста именования; с помощью ссылки поставщика, которая выражается с помощью значения атрибута **supplierKnowledge**, который хранится DSA теневого потребителя и который связан с контекстным префиксом для имени контекста. Оба этих атрибута содержатся в DSE с типом **cp**. Значение каждого из этих атрибутов создается при образовании теневого операционного связывания и обновляется при изменении теневого операционного связывания.

DSA поставщика может получить информацию для конструирования значения атрибута **secondaryShadows** в том случае, если необязательный компонент **secondaryShadows** для **ShadowingAgreementInfo** с потребителем равен **TRUE**. В этом случае всегда, когда DSA потребителя обнаруживает, что набор DSA, которые содержат копии повсеместно используемой дублируемой области (ее потребители или, в свою очередь, потребители ее потребителей и так далее, в зависимости от той глубины, для которой осуществляется операция вторичной теневого копирования) изменился (путем добавления, изменения или удаления точек доступа), он сообщает эту новую информацию (набор из **SupplierAndConsumers**) с помощью операции **modifyOperationalBinding**, как это описывается в Рек. МСЭ-Т X.525 | ИСО/МЭК 9594-9.

DSA поставщика поддерживает свой атрибут **secondaryShadows**, связанный с контекстным префиксом, следующим образом:

a) Набор **SupplierAndConsumers**, который был получен от потребителя с помощью операции **modifyOperationalBinding**, может использоваться для создания или замены значений атрибута. Компонент поставщика **SupplierAndConsumers** представляет точку доступа для DSA потребителя (или его потребителей, в зависимости от глубины вторичного теневого копирования); компонент потребителя – набор потребителей потребителя (или их потребителей, в зависимости от глубины вторичного теневого копирования).

b) Каждый потребитель, который обеспечивает своему поставщику операцию **modifyOperationalBinding**, которая содержит набор **SupplierAndConsumers**, включает следующие значения: значения его атрибута **secondaryShadows**, а также вновь сконструированное значение. Это значение конструируется из собственной точки доступа **myAccessPoint** (как компонента поставщика), а также значения точек доступа потребителя, которые содержатся в атрибуте **consumerKnowledge**, представляющем потребителей, хранящих широко используемых тени (в качестве компонентов потребителей).

Рекурсивное использование данной процедуры позволяет главному DSA для контекста именования знать обо всех вторичных теневых потребительских DSA, хранящих повсеместно используемые дублируемые области, выведенные от контекста именования. Данная информация затем доступна для поддержания подчиненных, неспецифических подчиненных и непосредственно вышестоящих ссылок.

23.1.2 Поддержание подчиненного и непосредственно вышестоящего знания в главных DSA

Подчиненная ссылка выражается с помощью значения атрибута **specificKnowledge**, который хранится в DSE типа **subr** с помощью того DSA, который содержит контекст именования для непосредственно вышестоящего, на который приводится ссылка. Непосредственная вышестоящая ссылка выражается через значение атрибута **specificKnowledge**, который хранится в DSE типа **immSupr** с помощью того DSA, который содержит контекст именования для непосредственно вышестоящего, на который приводится ссылка. Значение каждого из этих атрибутов создается в вышестоящих и подчиненных главных DSA при создании НОВ и обновляется при изменении НОВ.

Подчиненный главный DSA предоставляет вышестоящему главному DSA информацию для конструирования его подчиненной ссылки с помощью компонента **accessPoints** из параметра **SubordinateToSuperior**, который передается вышестоящему с помощью DOP. Информация, которая включается в **accessPoints**, определяется с помощью значений содержащихся в подчиненном DSA атрибутов следующим образом:

a) значение атрибута **myAccessPoint** (который содержится в корневом DSE) используется для формирования элемента в **accessPoints** с категорией **category** и значением **master**.

b) значения **consumerKnowledge** и **secondaryShadows** (они содержатся в подчиненном контекстном префиксе DSE) используются для формирования дополнительных элементов в **accessPoints** с категорией **category** и значением **shadow**.

Вышестоящий главный DSA обеспечивает нижестоящему главному DSA информацию для конструирования его непосредственно вышестоящей ссылки с помощью компонента **contextPrefixInfo** параметра **SuperiorToSubordinate**, который он передает подчиненному с помощью DOP. Данный компонент имеет значение типа **SEQUENCE OF Vertex**, которое содержит последовательность компонентов, соответствующих пути от корня дерева DIT к нижестоящему контекстному префиксу. Для одного из этих элементов, который соответствует контекстному префиксу непосредственно вышестоящего контекста именования, будет присутствовать необязательный компонент **accessPoints**. Подчиненный DSA содержит данную информацию как атрибут **specificKnowledge** в DSE типа **immSupr**, который соответствует этому же элементу **contextPrefixInfo**. Информация, которая включается в **accessPoints** вышестоящим DSA, определяется с помощью значений атрибутов вышестоящего DSA следующим образом:

a) Значение атрибута **myAccessPoint** (который содержится в корневом DSE) используется для формирования элемента в **accessPoints**, для которого значение **category** равно значению **master**.

b) Значения **consumerKnowledge** и **secondaryShadows** (содержатся в вышестоящем контекстном префиксе DSE) используются для формирования дополнительных элементов в **accessPoints**, для которых значение **category** равно значению **shadow**.

ПРИМЕЧАНИЕ. – Только те точки доступа, которые соответствуют DSA потребителя, получающего повсеместно используемые дублируемые области, должны выбираться вышестоящими и подчиненными DSA из своих атрибутов **consumerKnowledge** для включения в **accessPoints**. Эти процедуры, предназначенные для конструирования **secondaryShadows**, гарантируют, что данные точки доступа будут идентифицировать теневые DSA, содержащие повсеместно используемые дублируемые области.

23.1.3 Поддержание подчиненного и непосредственно вышестоящего знания в DSA потребителя

Теневой DSA потребителя, который договаривается со своим поставщиком о получении связанных с единицей репликации (копирования) знаний о непосредственно вышестоящем и подчиненном, на самом деле договаривается о том, что ссылки на непосредственного вышестоящего и подчиненного будут поддерживаться его DSA теневого поставщика посредством DISP.

ПРИМЕЧАНИЕ. – Для некоторых единиц спецификации репликации для DSA потребителя может понадобиться договориться о получении **extendedKnowledge** с той целью, чтобы поставщик мог предоставить знания о подчиненном.

23.2 Запрос перекрестных ссылок

Для того чтобы улучшить производительность системы Справочника, с помощью обыкновенных операций Справочника может быть расширен локальный набор перекрестных ссылок. Если DSA поддерживает DSP, то он может запросить другой DSA (который также поддерживает DSP) возвратить те ссылки на знания, которые содержат информацию о нахождении контекстов именования, связанных с названием целевого объекта обыкновенной операции Справочника.

Если компонент **returnCrossRefs** в **ChainingArguments** равен **TRUE**, то может присутствовать компонент **crossReferences** из **ChainingResults**, который представляет собой последовательность элементов перекрестных ссылок.

Если DSA не способен выполнить связывание запроса со следующим DSA, то инициировавшему DSA возвращается реферал. Если компонент **returnCrossRefs** из **ChainingArguments** равен **TRUE**, то реферал может дополнительно содержать контекстный префикс для контекста именования, на который ссылается реферал. Компонент **contextPrefix** отсутствует в том случае, если реферал основывается на неспецифической ссылке на подчиненного. Возвращаемая рефералом перекрестная ссылка основывается на знании, которое хранится в DSA, создавшем данный реферал.

В обоих этих случаях (связывание результатов и реферал) административный орган с помощью своего DSA может выбрать игнорирование запроса о возвращении перекрестных ссылок.

23.3 Несоответствия знания

Справочник должен поддерживать механизмы проверки соответствия с целью гарантировать в известной мере соответствие знания.

ПРИМЕЧАНИЕ. – При определенных обстоятельствах ссылка на знание будет являться точной (не являться недействительной в том смысле, который используется ниже), однако являться непригодной для использования DSA по той причине, что DMD рассматриваемого DSA в принципе не хочет проводить контакты с использованием ссылок на DSA (например, DSA который каким-то образом приобрел перекрестную ссылку на рассматриваемый DSA) или же не хочет контактировать в данной роли (например, в качестве главного DSA для контекста именования).

23.3.1 Обнаружение несоответствий знания

Вид несоответствия и способ его обнаружения различается для различных типов ссылок на знания:

a) *Перекрестные ссылки и подчиненные ссылки* – Данный тип ссылки является недействительным в том случае, если рассматриваемый DSA не содержит контекст именования или дублируемую область, полученную из контекста именования с контекстным префиксом, содержащимся в ссылке. Это несоответствие обнаруживается в процессе разрешения имени с помощью проверки компонентов **operationProgress** и **referenceType** из **ChainingArguments**.

b) *Неспецифические подчиненные ссылки* – Данный тип ссылки является недействительным в том случае, если рассматриваемый DSA не содержит локальный контекст именования с контекстным префиксом, который содержится в ссылке за вычетом последнего RDN. Для проверки соответствия используется описанный выше способ.

c) *Вышестоящие ссылки* – Недействительной вышестоящей ссылкой является такая, которая не является частью эталонного пути к корню. Поддержание вышестоящих ссылок должно обеспечиваться внешними средствами и не является предметом данной спецификации Справочника.

   ПРИМЕЧАНИЕ. – Не всегда возможно обнаружить недействительную вышестоящую ссылку.

d) *Непосредственные вышестоящие ссылки* – Данный тип ссылок является недействительным в том случае, если рассматриваемый DSA не содержит контекст именования или копируемую область, полученную от контекстного префикса, содержащегося в ссылке. Далее, использование данного типа ссылок является действительным только тогда, когда компонент **operationProgress** из **ChainingArguments** имеет значение **notStarted** или **proceeding**. Данное несоответствие обнаруживается в процессе разрешения имени с помощью проверки компонентов **operationProgress** и **referenceType** из **ChainingArguments**.

e) *Ссылки на поставщика* – Данный тип ссылки, которая указывает на поставщика копируемой области и, в качестве опции, на мастера контекста именования, от которого получена копируемая область, является недействительным в том случае, если рассматриваемый DSA не является теневым поставщиком для DSA, который использует данную ссылку (когда компонент **referenceType** из **ChainingArguments** имеет значение **supplier**) или если рассматриваемый DSA не является мастером для контекста именования (когда **referenceType** имеет значение **master**). Данное несоответствие обнаруживается при выполнении фазы разрешения имени и фазы оценки для данной операции при помощи проверки компонента **referenceType** из **ChainingArguments**.

23.3.2 Отчет о несоответствии знания

Если при выполнении запроса Справочника используется связывание, то все несоответствия знания будут обнаружены DSA, который содержит недействительную ссылку на знание, при помощи получения ошибки **serviceError** с проблемой **invalidReference**.

Если DSA возвращает реферал, который основывается на недействительной ссылке на знание, то заказчику возвращается ошибка **serviceError** с проблемой **invalidReference** при том условии, что он использует реферал. То, каким образом условия ошибки распространяется к тому DSA, который хранит недействительную ссылку, не является предметом данной спецификации Справочника.

23.3.3 Обработка несоответствующих ссылок на знания

После того, как DSA обнаружил недействительную ссылку, он должен восстановить соответствие знания. Например, это может быть сделано с помощью простого удаления недействительной перекрестной ссылки или же с помощью ее замены на корректную перекрестную ссылку, которая может быть получена с помощью механизма **returnCrossRefs**.

Способ, с помощью которого DSA действительно обрабатывает недействительную ссылку, является локальным вопросом и не относится к данной спецификации Справочника.

23.4 Ссылки на знания и контексты

Названия в ссылке знания должны являться первичными выделенными именами и также могут включать альтернативные выделенные значения и информацию о контексте, которая хранится в **valuesWithContext** для любого атрибута, вносящего вклад в некое имя RDN, как это описывается в п. 9.3 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

В зависимости от того, каким образом получена ссылка на знание (в особенности тогда, когда ссылка содержится в DSA с изданием ниже третьего или же такой DSA является частью пути, с помощью которого была получена данная ссылка), существует возможность, что ссылка знания не будет включена во все возможные альтернативные выделенные имена. Это может привести к тому, что потенциальное имя не будет распознано как то же имя держателем ссылки на знания, что приводит к дополнительным шагам при

разрешении имени или, в некоторых ситуациях, к получению несоответствующих результатов или неудаче при разрешении имени. Всеобщее использование первичных выделенных имен там, где они известны, оптимизирует возможности Справочника по обращению с содержащимися в именах вариантами контекста.

24 Иерархические операционные связывания

Иерархическое операционное связывание используется для представления отношений между двумя DSA, которые содержат два контекста именования, один из которых является непосредственно подчиненным для другого. В случае HOB вышестоящий DSA содержит подчиненную ссылку на контекст именования, который содержится подчиненным DSA; подчиненный DSA содержит непосредственно вышестоящую ссылку на контекст именования, который содержит вышестоящий DSA. Операционное связывание обеспечивает, что соответствующая информация знания обменивается и поддерживается между двумя DSA таким образом, что два DSA имеют возможность вести себя в процессе разрешения имени и оценки операции так, как это определяется в пп. 18 и 19.

24.1 Характеристики типа операционного связывания

24.1.1 Симметрия и роли

Тип иерархического операционного связывания является ассиметричным типом операционного связывания. Две роли в связывании данного типа являются следующими:

a) роль главного DSA для вышестоящего контекста именования – *superior DSA* (ассоциируется с абстрактной ролью "A"); и

b) роль главного DSA для подчиненного контекста именования – *subordinate DSA* (ассоциируется с абстрактной ролью "B").

24.1.2 Соглашение

Информация о соглашении, которой обмениваются при установлении иерархического операционного связывания, является значением **HierarchicalAgreement**. Здесь содержится относительное выделенное имя для нового контекстного префикса (компонент **rdn**) и выделенное имя для статьи, которая является непосредственно вышестоящей для нового контекста именования (компонент **immediateSuperior**). Данная информация предоставляется тем DSA, который инициирует HOB.

```
HierarchicalAgreement::= SEQUENCE {
        rdn                 [0]   RelativeDistinguishedName,
        immediateSuperior   [1]   DistinguishedName }
```

Компонент **rdn** должен являться первичным RDN, а компонент **immediateSuperior** должен являться первичным выделенным именем. Информация о контексте и все альтернативные выделенные значения также должны включаться в компонент **valuesWithContext** из **AttributeTypeAndDistinguishedValue** для любого RDN, как это описывается в п. 9.3 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

24.1.3 Инициатор

24.1.3.1 Образование

Образование иерархического операционного связывания может инициироваться каждой из ролей. Инициирование со стороны вышестоящего DSA может быть вызвано операцией Add Entry (добавление статьи), где подчиненный DSA специфицируется в расширении **targetSystem**, или же с помощью административного вмешательства. Инициирование со стороны подчиненного DSA (который соединяет существующую локально статью или поддерево с глобальным деревом DIT) вызывается административным вмешательством.

24.1.3.2 Изменение

Изменение иерархического операционного связывания может быть инициировано каждой из ролей. Вышестоящий DSA может выдать изменение в результате изменения информации о вышестоящем контекстном префиксе. Это может являться результатом любой операции изменения или же результатом административного вмешательства.

Каждый DSA может изменить соглашение в результате модификации RDN для статьи контекстного префикса нижестоящего контекста именования. Вышестоящий DSA инициирует данное изменение, так как было изменении относительное выделенное имя, расположенное выше в дереве DIT, или же по причине административного вмешательства. Подчиненный DSA инициирует изменение по причине ModifyDN для контекстного префикса или же по причине административного вмешательства.

Также каждый DSA может изменять HOB в том случае, если изменяется информация точки доступа для его контекста именования.

24.1.3.3 Завершение

Завершение иерархического операционного связывания может быть инициировано каждой из ролей. Инициация вышестоящим DSA может быть вызвана административным вмешательством. Инициация подчиненным DSA может быть вызвана либо операцией удаления статьи (Remove Entry), которая удаляет статью контекстного префикса для подчиненного контекста именования, или же административным вмешательством.

24.1.4 Параметры образования

Параметры образования различаются для двух ролей НОВ – вышестоящего DSA и подчиненного DSA. Параметром образования для роли вышестоящего DSA является значение **SuperiorToSubordinate**, параметром для подчиненной роли является значение **SubordinateToSuperior**.

24.1.4.1 Параметры образования для вышестоящего DSA

Параметр учреждения, который выдается вышестоящим DSA, значение **SuperiorToSubordinate**, обеспечивает подчиненному DSA информацию относительно вершин дерева DIT, которые являются вышестоящими для контекстных префиксов нового контекста именования (который включает непосредственную вышестоящую ссылку) и также в качестве опции пользовательские и операционные атрибуты для статьи подчиненного контекстного префикса и копии пользовательских и операционных атрибутов для статьи, которая является непосредственно вышестоящей для нового контекстного префикса.

```
SuperiorToSubordinate::= SEQUENCE {
        contextPrefixInfo           [0]  DITcontext,
        entryInfo                   [1]  SET SIZE (1..MAX) OF Attribute OPTIONAL,
        immediateSuperiorInfo       [2]  SET SIZE (1..MAX) OF Attribute OPTIONAL }
```

Компонент **rdn** в **Vertex** или в **SubentryInfo** должен являться первичным RDN, а контекстная информация и все выделенные значения должны включаться в компоненты **AttributeTypeAndDistinguishedValue** для RDN, как это описывается в п. 9.3 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

24.1.4.1.1 Информация контекстного префикса

Компонент **contextPrefixInfo** из **SuperiorToSubordinate** является значением типа **DITcontext**, который представляет собой последовательность значений **Vertex**.

```
DITcontext::= SEQUENCE OF Vertex

Vertex::= SEQUENCE {
        rdn             [0]  RelativeDistinguishedName,
        admPointInfo    [1]  SET SIZE (1..MAX) OF Attribute OPTIONAL,
        subentries      [2]  SET SIZE (1..MAX) OF SubentryInfo OPTIONAL,
        accessPoints    [3]  MasterAndShadowAccessPoints OPTIONAL }
```

Компонент **contextPrefixInfo** является последовательностью RDN, которые формируют выделенное имя непосредственно вышестоящего для нового контекстного префикса, каждый RDN (который определяется компонентом **rdn**) также может в качестве опции сопровождаться дополнительной информацией.

Необязательный компонент **admPointInfo** из **Vertex** сигнализирует о том, что вершина дерева DIT является административной точкой и обеспечивает, по меньшей мере, собственный операционный атрибут **administrativeRole**.

Информация подстатьи, которая связана с административной точкой, обеспечивается компонентом **subentries** из **Vertex**, который представляет собой набор из одного или более значений **SubentryInfo**. Каждое значение **SubentryInfo** состоит из RDN данной подстатьи (компонент **rdn**) и атрибутов подстатьи (компонент **info**).

```
SubentryInfo::= SEQUENCE {
        rdn     [0]  RelativeDistinguishedName,
        info    [1]  SET OF Attribute }
```

Необязательный компонент **accessPoints** из **Vertex** сигнализирует о том, что вершина соответствует контекстному префиксу для непосредственно вышестоящего контекста именования. Вышестоящий использует этот компонент для того, чтобы обеспечить подчиненному информацию, которая необходима для его непосредственно подчиненной ссылки.

ПРИМЕЧАНИЕ. – Главная точка доступа в пределах **accessPoints** является той же, что и значение, переданное в параметре **accessPoint** для операций образования или изменения операционного иерархического связывания.

24.1.4.1.2 Информация о статье

Необязательный компонент **entryInfo** из **SuperiorToSubordinate** представляет собой набор атрибутов, устанавливающих содержание новой статьи контекстного префикса.

24.1.4.1.3 Информация о непосредственно вышестоящей записи

Необязательный компонент **immediateSuperiorInfo** из **SuperiorToSubordinate** является копией набора атрибутов, в частности **objectClass** и **entryACI**, относящихся к статье, непосредственно вышестоящей для нового контекстного префикса.

ПРИМЕЧАНИЕ. – Данный компонент может использоваться подчиненным для оптимизации оценки запроса List (список), который генерирует пустой список **ListResult** для базового объекта, который является непосредственным вышестоящим для подчиненного контекстного префикса [см. Примечание в п. 19.3.1.2.2, подпункт 2)].

24.1.4.2 Параметр образования подчиненного DSA

Параметр образования, который выдается подчиненным DSA и является значением **SubordinateToSuperior**, обеспечивает вышестоящему DSA информацию относительно подчиненного контекста именования.

```
SubordinateToSuperior::=    SEQUENCE {
      accessPoints    [0]   MasterAndShadowAccessPoints OPTIONAL,
      alias           [1]   BOOLEAN DEFAULT FALSE,
      entryInfo       [2]   SET SIZE (1..MAX) OF Attribute OPTIONAL,
      subentries      [3]   SET SIZE (1..MAX) OF SubentryInfo OPTIONAL }
```

Компонент **accessPoints** из **SubordinateToSuperior** используется подчиненным для того, чтобы обеспечить вышестоящего информацией, необходимой для его подчиненной ссылки.

ПРИМЕЧАНИЕ 1. – Главная точка доступа в пределах **accessPoints** является той же, что и значение, переданное в параметре **accessPoint** для операций образования или изменения операционного иерархического связывания.

Компонент **alias** из **SubordinateToSuperior** используется для того, чтобы сообщить вышестоящему о том, что подчиненный контекст именования состоит из единственной статьи псевдонима.

Компонент **entryInfo** из **SubordinateToSuperior** состоит из копии набора атрибутов, в частности **objectClass** и **entryACI**, а также, если это применимо, операционного атрибута **administrativeRole** из статьи нового контекстного префикса.

ПРИМЕЧАНИЕ 2. – Первые два атрибута могут использоваться вышестоящим для оптимизации оценки запроса List (список) или одноуровневого запроса Search (поиск), базовым объектом которого является непосредственно вышестоящий для подчиненного контекстного префикса, в то время как последний атрибут используется для того, чтобы избежать нежелательного продвижения операции поиска внутрь или вне специфической для службы административной области.

Компонент **subentries** из **SubordinateToSuperior** используется подчиненным для передачи подзаписей, содержащих предписывающие ACI для вышестоящего.

24.1.5 Параметры изменения

Для изменения HOB параметр изменения **SuperiorToSubordinateModification** в случае вышестоящей роли равен **SuperiorToSubordinate**, с тем ограничением, что компонент **entryInfo** может отсутствовать; в случае подчиненной роли равен **SubordinateToSuperior**.

```
SuperiorToSubordinateModification::= SuperiorToSubordinate (
        WITH COMPONENTS {..., entryInfo ABSENT})
```

Эти параметры идентичны (с учетом указанных выше ограничений) соответствующим параметрам образования и используются для того, чтобы сообщить об изменениях, которые происходят с информацией, предоставленной в параметрах образования вслед за образованием HOB.

Если каждый компонент **SuperiorToSubordinate** (или последовательно **SuperiorToSubordinateModification**) или **SubordinateToSuperior** испытывает изменения (например, компонента **contextPrefixInfo** из **SuperiorToSubordinate**), то должен быть обеспечен соответствующий компонент параметра изменения (например, компонент **contextPrefixInfo** из **SuperiorToSubordinateModification**) в своей полной форме в изменении операционного связывания.

24.1.6 Параметры завершения

Ни одна из ролей не обеспечивает параметр завершения при завершении HOB.

24.1.7 Идентификация типа

Иерархическое операционное связывание определяется с помощью идентификатора объекта, который присваивается при определении информационного объекта **hierarchicalOperationalBinding OPERATIONAL-BINDING** в п. 24.2.

24.2 Определение класса Operational binding information object (информационный объект операционного связывания)

Данный подраздел определяет тип иерархического операционного связывания с использованием шаблона класса информационного объекта **OPERATIONAL-BINDING**, определенного в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

```
hierarchicalOperationalBinding OPERATIONAL-BINDING::= {
        AGREEMENT       HierarchicalAgreement
        APPLICATION CONTEXTS {
                        {directorySystemAC} }
        ASYMMETRIC
            ROLE-A {            -- вышестоящий DSA
                ESTABLISHMENT-INITIATOR         TRUE
                ESTABLISHMENT-PARAMETER         SuperiorToSubordinate
                MODIFICATION-INITIATOR          TRUE
                MODIFICATION-PARAMETER          SuperiorToSubordinateModification
```

```
        TERMINATION-INITIATOR           TRUE }
    ROLE-B {        -- подчиненный DSA
        ESTABLISHMENT-INITIATOR         TRUE
        ESTABLISHMENT-PARAMETER         SubordinateToSuperior
        MODIFICATION-INITIATOR          TRUE
        MODIFICATION-PARAMETER          SubordinateToSuperior
        TERMINATION-INITIATOR           TRUE }
ID              id-op-binding-hierarchical }
```

24.3 Процедуры DSA для управления иерархическим операционным связыванием

В последующих процедурах новый DSE или отметка (например, индикатор состояния, связанный с каким-то из пунктов информации), созданные DSA, должны быть помещены в надежное хранилище. Благодаря этому для двух DSA становится возможным с помощью показанных ниже процедур поддерживать совместимое толкование параметров для HOB в присутствии неисправностей коммуникаций и конечной системы.

В двух описываемых ниже процедурах **establishment** и **modification**, тот DSA, который выполняет роль отвечающей стороны (т. е. не тот, кто инициирует образование или изменение) может предоставить DSA, который играет роль инициатора информацию (например, операционные атрибуты), которая какой-то причине является неприемлемой. В таким случаях инициирующий DSA может завершить операционное связывание.

24.3.1 Процедура образования

24.3.1.1 Образование, инициированное вышестоящим DSA

Если DSA выполняет оценку операции Add Entry (добавить статью) с другим DSA, который указывается в расширении **targetSystem**, то он должен установить иерархическое операционное связывание в соответствии со следующей процедурой. Если по административным причинам DSA хочет установить HOB с подчиненным DSA, и он поддерживает протокол DOP HOB, то выполняется следующая процедура:

1) Вышестоящий DSA создает новый DSE с типом **subr** с названием новой статьи и отмечает этот новый DSE как *being added* (*добавлен*). Вышестоящий DSA генерирует уникальный идентификатор связывания **bindingID** и сохраняет его в новом DSE.
2) Вышестоящий DSA должен послать подчиненному DSA операцию образования операционного связывания (Establish Operational Binding), которая содержит следующие параметры:
   a) **bindingType** устанавливается как **hierarchicalOperationalBindingID**;
   b) в параметре образования **SuperiorToSubordinate** присутствуют компоненты **contextPrefixInfo** и **entryInfo**; все другие параметры являются необязательными;
   c) **HierarchicalAgreement** с компонентом **immediateSuperior** устанавливается равным выделенному имени непосредственного вышестоящего для новой статьи и компонент **rdn** устанавливается как RDN новой статьи;
   d) параметры **bindingID**, **myAccessPoint** и **valid** устанавливаются соответствующим образом.
3) Если подчиненный DSA принимает операцию, то он создает (соответствующим образом) требуемые DSE с типами **glue**, **subentry**, **admPoint**, **rhob** и **immSupr,** которые представляют **contextPrefixInfo**; DSE типа **cp** и **entry** или **alias**, которые представляют новый объект контекстного префикса или статью псевдонима; и, в соответствующих случаях, DSE типа **rhob** и **entry** для представления **immediateSuperiorInfo**. Он сохраняет **bindingID** с DSE нового контекстного префикса и возвращает параметр **SubordinateToSuperior** вышестоящему DSA.

   Если подчиненный DSA отказывается от операции, то он возвращает ошибку операционного связывания с установленным значением для соответствующей проблемы.

   Если контекст именования уже существует и значения **bindingID** для существующего и нового контекста одинаковы, то подчиненный DSA уже создал запрашиваемый контекст именования и в этом случае нижестоящий DSA возвращает результат вышестоящему. Если значения не равны, то передается ошибка операционного связывания с проблемой **invalidAgreement**; это означает, что вышестоящий DSA имеет постоянное несоответствие в знании, которое требует корректировки со стороны администратора.
4) Если вышестоящий DSA получает ошибку, то он удаляет отмеченный DSE типа **subr** и возвращает ошибку для операции Add Entry (добавить статью).

   Если вышестоящий DSA получает результат, то он удаляет отметку с DSE, который представляет **subr**, и возвращает результат для операции Add Entry.

   Если случается какая-то неисправность (например, неисправность коммуникаций или конечной системы), то вышестоящий DSA должен повторять этапы начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего исполнения образования иерархического операционного связывания, для которого он выступает в роли инициатора. Если образование является результатом операции Add Entry (добавить статью), и запрашивающая сторона прекращает операцию (например, освобождая или прекращая прикладную ассоциацию) до того, как образование будет завершено, то вышестоящий DSA должен игнорировать данное событие и должен завершить образование (которое может завершиться успехом или неудачей).

В этом случае пользователь не будет проинформирован о результате выполнения операции Add Entry (добавить статью).

ПРИМЕЧАНИЕ 1. – Отметка подчиненных помогает при восстановлении и контроле согласованности по времени. Другой пользователь не может добавлять статью, которая уже отмечена, и DSA после неудачной попытки повторяет образование операционного связывания для всех выделенных подчиненных.

ПРИМЕЧАНИЕ 2. – Для показанной выше процедуры несоответствие для знания может носить только временный характер. То, как вышестоящий DSA обрабатывает несвязанные между собой операции, которые выполняют чтение подчиненных ссылок при наличии отметки, является локальным вопросом.

24.3.1.2 Образование, инициированное подчиненным DSA

Подчиненный DSA может инициировать иерархическое операционное связывание. Это может происходить в результате желания администратора связать содержащееся в DSA поддерево статей с определенной точкой глобального дерева DIT. В этом случае подчиненный DSA должен установить HOB в соответствии со следующей процедурой:

1) Подчиненный DSA либо уже имеет DSE типа **cp** в качестве части существующего контекста именования или же он создает новый. Он отмечает DSE как *being added* (*добавлен*), генерирует уникальный идентификатор **bindingID** и сохраняет его вместе с контекстным префиксом DSE.
2) Подчиненный DSA посылает вышестоящему DSA операцию образования операционного связывания (Establish Operational Binding) со следующими параметрами:
   a) **bindingType** устанавливается как **hierarchicalOperationalBindingID**;
   b) параметр образования **SubordinateToSuperior** устанавливается соответствующим образом;
   c) **HierarchicalAgreement** с компонентом **immediateSuperior** устанавливается равным выделенному имени непосредственного вышестоящего новой статьи и компонент **rdn** устанавливается равным RDN новой статьи;
   d) параметры **bindingID**, **myAccessPoint** и **valid** устанавливаются соответствующим образом.

   Если вышестоящий DSA отказывается от операции, то он возвращает ошибку операционного связывания (Operational Binding Error) с установленным значением для соответствующей проблемы.
3) Вышестоящий DSA проверяет, что он является мастером для непосредственного вышестоящего для статьи нового контекстного префикса или возвращает ошибку **operationalBindingError** с проблемой **roleAssignment**.
4) Вышестоящий DSA проверяет, что запрашиваемый RDN для нового контекстного префикса пока не используется. Если сопоставление для RDN не обнаруживается с помощью информации, хранимой локально, однако непосредственно вышестоящий DSE относится к типу **nssr**, то выполняется процедура из п. 19.1.5. Если данная процедура также не обнаруживает сопоставления для RDN, то вышестоящий DSA создает DSE с типом **subr**, сохраняет вместе с ним **bindingID** и возвращает результат.

   Если в данном RDN обнаруживается подчиненная ссылка, то сравниваются два значения **bindingID**. Если эти значения равны, то возвращается результат. Параметр **SuperiorToSubordinate**, который возвращается вышестоящим DSA, не должен содержать компонент **entry**. Если же два значения **bindingID** не равны, то отправляется ошибка **operationalBindingError** с проблемой **invalidAgreement**; это означает, что вышестоящий DSA имеет постоянное несоответствие знания, которое требует корректировки со стороны администратора.

   Если при поиске в NSSR обнаруживается сопоставление для RDN, то посылается ошибка **operationalBindingError** с проблемой **invalidAgreement**; это также означает, что вышестоящий DSA имеет постоянное несоответствие знания, которое требует корректировки со стороны администратора.
5) Если подчиненный DSA получает ошибку, то он удаляет новый контекстный префикс DSE и его отметку. То, как должна обрабатываться информация статьи, от которой был получен контекстный префикс DSE, является локальным вопросом.

   Если подчиненный DSA получает результат, он соответствующим образом добавляет необходимые DSE с типами **glue**, **subentry**, **admPoint**, **rhob** и **immSupr**, которые представляет **contextPrefixInfo**; и, по обстоятельствам, DSE типа **rhob** и **entry**, которые должен представлять **immediateSuperiorInfo**. Отметка контекстного префикса DSE удаляется.

   Если происходит какая-то неисправность (например, неисправности коммуникаций или конечной системы), то подчиненный DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будет получен результат или ошибка для каждого ожидающего образования иерархического операционного связывания, для которого он является инициатором.

24.3.2 Процедура изменения

Для изменения HOB определяются следующие процедуры, которые инициируются процедурой, описанной в п. 24.3.1.

24.3.2.1 Процедура изменения, которая инициируется вышестоящим

Данная процедура может вызываться в результате операций изменения, как это описывается в п. 19.1, или же в результате административного вмешательства (например, для передачи изменений параметрам **myAccessPoint**, **agreement** или **valid** для данной HOB). Также если вышестоящий DSA обнаруживает изменения в компонентах

contextPrefixInfo или **immediateSuperiorInfo** для значения **SuperiorToSubordinate**, которое передается подчиненному DSA, то он должен передать новую информацию подчиненному DSA с использованием следующей процедуры:

1) Отмечаем DSE с типом **subr** как *being modified* (*изменено*), и если данное изменение является результатом изменения RDN статьи подчиненного контекстного префикса, то добавляется новый DSE типа **subr** и он отмечается как *being added* (*добавлен*).
2) Вышестоящий DSA создает новое значение **bindingID** из существующего значения путем инкрементирования компонента **version**. Используя новое значение **bindingID**, он посылает операцию изменения операционного связывания вышестоящему DSA с параметром изменения **SuperiorToSubordinateModification**.
3) Подчиненный DSA проверяет компонент **identifier** из **bindingID**. Если он не имеет такого соглашения с вышестоящим, или же компонент **version** имеет значение, меньшее чем версия НОВ, то он должен возвратить ошибку **operationalBindingError** с проблемой **invalidAgreement**.
4) Подчиненный DSA может принять изменение для НОВ, модифицировать или вновь создать DSE, которые представляют информацию контекстного префикса, обновить компонент **version** из собственного идентификатора **bindingID** и возвратить результат. В противном случае он может возвратить ошибку и затем завершить соглашение.
5) Если вышестоящий DSA получает результат, то изменение завершено. Если данное изменение является результатом изменения RDN для статьи подчиненного контекстного префикса, то для нового DSE с типом **subr** и отметкой *being added* (*добавлен*) удаляется его отметка, а старый DSE с отметкой *being modified* (*изменен*) удаляется. В противном случае просто удаляется отметка *being modified* (*изменен*).

   Если вышестоящий DSA получает ошибку, то изменение завершилось неудачей. Отметка *being modified* (*изменен*) удаляется. Если данное изменение является результатом изменения RDN для статьи подчиненного контекстного префикса, то удаляется новый DSE с типом **subr** и отметкой *being added* (*добавлен*). В противном случае предпринимаемые меры выходят за рамки данной спецификации Справочника.

   Если возникает какая-то неисправность (например, неисправность коммуникаций или конечной системы), то вышестоящий DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего своего исполнения изменения иерархического операционного связывания, для которого он является инициатором. Если изменение является результатом операции ModifyDN, которая изменяет RDN записи подчиненного контекстного префикса, и заказчик прекращает данную операцию (например, освобождая или прекращая прикладную ассоциацию) до завершения изменения, то вышестоящий DSA должен игнорировать данное событие и завершить изменение (которое может быть удачным или нет). В этом случае пользователь не будет проинформирован о результате операции ModifyDN.

24.3.2.2 Процедура изменения, инициированная подчиненным

Данная процедура может вызываться в результате административного вмешательства (например, чтобы передать изменения параметрам **myAccessPoint**, **agreement** или **valid**, относящимся к НОВ). Кроме этого, если подчиненный DSA обнаруживает изменения в значении **SubordinateToSuperior**, которое он передал вышестоящему DSA, то он должен передать новую информацию вышестоящему DSA с использованием следующей процедуры:

1) Отмечаем DSE с типом **cp** как *being modified* (*изменен*).
2) Подчиненный DSA создает новое значение **bindingID** из существующего значения путем инкрементирования компонента **version**. Используя новое значение **bindingID**, он отсылает операцию изменения операционного связывания вышестоящему DSA с параметром изменения **SubordinateToSuperior**.
3) Вышестоящий DSA проверяет компонент **identifier**, содержащийся в **bindingID**. Если у него не существует такого соглашения с подчиненным, или же компонент **version** имеет значение, меньшее чем версия НОВ, то он должен возвратить ошибку **operationalBindingError** с проблемой **invalidAgreement**.
4) Вышестоящий DSA может принять изменение НОВ, изменить DSE, который представляет подчиненную ссылку и возвратить результат. В противном случае он может возвратить ошибку и затем завершить соглашение.

   В дополнение к этому, если вышестоящий DSE для DSE (типа **subr**), который должен быть переименован, относится к типу **nssr**, то DSA должен действовать в соответствии с процедурой, определенной в п. 19.1.5 (Операции изменения и NSSR), для того чтобы обеспечить уникальность нового имени статьи и только после этого отвечать на запрос изменения НОВ.
5) Если подчиненный DSA получает результат, то изменение завершено и он удаляет отметку. Если же он получает ошибку, то предпринимаемые меры выходят за рамки данной спецификации Справочника.

   Если возникает какая-то неисправность (например, неисправность коммуникаций или конечной системы), то подчиненный DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего своего исполнения изменения иерархического операционного связывания, для которого он является инициатором.

24.3.3 Процедура завершения

Для завершения НОВ, который был инициирован процедурой из п. 24.3.1, определяются следующие процедуры.

24.3.3.1 Завершение вышестоящим DSA

Завершение иерархического операционного связывания инициируется вышестоящим DSA только в результате административного вмешательства. Необходимо следовать следующей процедуре:

1) Вышестоящий DSA отмечает DSE, который представляет подчиненную ссылку, как *being deleted* (*удален*), таким образом данная подчиненная ссылка уже не используется при разрешении имени.
2) Вышестоящий DSA посылает операцию Terminate Operational Binding (завершить операционное связывание) для иерархического операционного связывания с подчиненным DSA. Вышестоящий DSA при этом опускает компонент **version** из идентификатора **bindingID**.
3) Когда подчиненный DSA получает Terminate Operational Binding (завершить операционное связывание), то он удаляет всю информацию относительно иерархического операционного связывания и посылает результат, если только не является неизвестным компонент **identifier** из **bindingID**, в этом случае возвращается ошибка **operationalBindingError** с проблемой **invalidID**. Определение того, как должна обрабатываться информация записи, связанная с подчиненным контекстом именования, является локальным вопросом.
4) Если вышестоящий DSA получает результат или ошибку **operationalBindingError** с проблемой **invalidID**, то он должен удалить DSE с отметкой *being deleted* (*удален*), который представляет подчиненную ссылку, связанную с иерархическим операционным связыванием, и удаляет всю информацию об операционном связывании.

   Если происходит какая-то неисправность (например, неисправность коммуникаций или конечной системы), то вышестоящий DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будет получен результат или ошибка для каждого ожидающего выполнения завершения иерархического операционного связывания, для которого он является инициатором.

24.3.3.2 Завершение, инициированное подчиненным DSA

Завершение, которое инициируется подчиненным DSA, может вызываться операцией Remove Entry (удаление статьи), которая удаляет последнюю статью в пределах подчиненного контекста именования или статью контекстного префикса, или же может также вызываться административным вмешательством. Необходимо следовать следующей процедуре:

1) Подчиненный DSA отмечает DSE контекстного префикса для контекста именования как *being deleted* (*удалено*).
2) Подчиненный DSA посылает операцию Terminate Operational Binding (завершение иерархического связывания) иерархическому операционному связыванию с вышестоящим DSA. Подчиненный DSA при этом опускает компонент **version** из идентификатора **bindingID**.
3) Когда вышестоящий DSA получает Terminate Operational Binding (завершить операционное связывание), то он удаляет DSE, который представляет подчиненную ссылку, связанную с иерархическим операционным связыванием, удаляет всю информацию относительно иерархического операционного связывания и посылает результат, если только не является неизвестным компонент **identifier** из **bindingID.** В этом случае возвращается ошибка **operationalBindingError** с проблемой **invalidID**.
4) Если подчиненный DSA получает результат или ошибку **operationalBindingError** с проблемой **invalidID**, то он должен удалить всю информацию об операционном связывании.

   ПРИМЕЧАНИЕ. – То, как обрабатывается далее информация статьи контекста именования, является локальным вопросом для подчиненного DSA. Так как переименование (т. е. перемещение) контекста именования не разрешено операцией Modify DN, то администратор может, например, завершить НОВ, выбрать другой контекстный префикс для контекста именования и вновь соединить его с другой частью дерева DIT (т. е. установить новое НОВ).

   Если происходит какая-то неисправность (например, неисправность коммуникаций или конечной системы), то подчиненный DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будет получен результат или ошибка для каждого ожидающего выполнения завершения иерархического операционного связывания, для которого он является инициатором.

24.4 Процедуры для операций

Операции, которые могут выполняться в кооперативном состоянии для иерархического операционного связывания, это те, которые определены в контексте применения **directorySystemAC**.

Процедуры, которым должны следовать DSA, участвующие в иерархическом операционном связывании, должны соответствовать тем, что определены в пп. 16–22.

24.5 Использование контекстов применения

Для того чтобы образовать, изменить или завершить иерархическое операционное связывание с использованием протокола и процедур данного стандарта Справочника, DSA должен использовать контекст применения **operationalBindingManagementAC**.

25 Неспецифическое иерархическое операционное связывание

Неспецифическое операционное связывание используется для представления отношений между двумя DSA, которые содержат два контекста именования, один из которых является непосредственно подчиненным для другого. В случае NHOB вышестоящий DSA содержит неспецифическую подчиненную ссылку на контекст именования, который содержится в подчиненном DSA, при этом вышестоящий DSA содержит непосредственную вышестоящую ссылку на контекст именования, который содержится в вышестоящем DSA.

операционное связывание гарантирует, что соответствующая информация знания обменивается и поддерживается между двумя DSA, так что оба DSA способны действовать в процессе разрешения имени и оценки операции так, как это определено в пп. 18 и 19.

25.1 Характеристики типа операционного связывания

25.1.1 Симметрия и роли

Тип иерархического операционного связывания относится к несимметричному типу операционного связывания. При связывании такого типа существует две различные роли:

a) роль мастера DSA для вышестоящего контекста именования – *superior DSA* (ассоциируется с абстрактной ролью "A"); и

b) роль мастера DSA для подчиненного контекста именования – *subordinate DSA* (ассоциируется с абстрактной ролью "B").

25.1.2 Соглашение

Информация соглашения, которой обмениваются при образовании неспецифического операционного связывания, является значением **NonSpecificHierarchicalAgreement** и содержит только выделенное имя статьи, непосредственно вышестоящей для нового контекста именования (компонент **immediateSuperior**). Эта информация должна предоставляться тем DSA, который инициировал данное NHOB.

```
NonSpecificHierarchicalAgreement::= SEQUENCE {
        immediateSuperior      [1]   DistinguishedName }
```

ПРИМЕЧАНИЕ. – То, каким образом подчиненный DSA определяет, что имя нового контекста именования является уникальным, выходит за рамки данной Рекомендации | Международного стандарта. Имя является уникальным, если оно корректно присваивается соответствующим органом по именованию и никакой другой DSA не содержит это же имя в качестве главной статьи.

25.1.3 Инициатор

25.1.3.1 Образование

Образование неспецифического иерархического операционного связывания может быть инициировано только ролью подчиненного DSA. Инициация подчиненным DSA (который соединяет одну или более существующих локально записей или поддеревьев с глобальным деревом DIT) происходит под влиянием административного вмешательства.

25.1.3.2 Изменение

Изменение неспецифического иерархического связывания может быть вызвано каждой из ролей. Вышестоящий DSA может вызвать изменение в результате изменения информации вышестоящего контекстного префикса. Это может происходить как в результате любой из операций изменения, так и в результате административного вмешательства.

Каждый из DSA также может изменить NHOB, если изменяется информация точки доступа для его контекста именования (или в случае подчиненной роли один из непосредственно подчиненных контекстов именования).

25.1.3.3 Завершение

Завершение иерархического операционного связывания может инициироваться каждой из ролей. Инициация вышестоящим DSA может быть вызвана административным вмешательством. Инициация подчиненным DSA может быть вызвана либо операцией Remove Entry (удалить запись) для удаления конечной статьи контекстного префикса, которая хранится подчиненным, являющимся непосредственно подчиненным для компонента **immediateSuperior** соглашения, либо может быть вызвана административным вмешательством.

25.1.4 Параметры образования

Параметр образования, который выпускается вышестоящим DSA, является значением **NHOBSuperiorToSubordinate** и эквивалентен соответствующему параметру образования HOB за тем исключением, что отсутствует компонент **entryInfo**.

```
NHOBSuperiorToSubordinate::= SuperiorToSubordinate (
        WITH COMPONENTS {..., entryInfo ABSENT})
```

Параметр образования, который выпускается подчиненным DSA, является значением **NHOBSubordinateToSuperior** и эквивалентен соответствующему параметру установления HOB за тем исключением, что отсутствуют компоненты **alias** и **entryInfo**.

```
NHOBSubordinateToSuperior::= SEQUENCE {
    accessPoints      [0]   MasterAndShadowAccessPoints OPTIONAL,
    subentries        [3]   SET SIZE (1..MAX) OF SubentryInfo OPTIONAL }
```

25.1.5 Параметры изменения

Данные параметры идентичны соответствующим параметрам образования и используются для того, чтобы сигнализировать об изменениях в информации, предоставленной в параметрах образования вслед за образованием NHOB.

Если происходит изменение в каком-либо компоненте из **NHOBSuperiorToSubordinate** или **NHOBSubordinateToSuperior** (например, в компоненте **contextPrefixInfo** из **NHOBSuperiorToSubordinate**), то соответствующий компонент параметра изменения (например, компонент **contextPrefixInfo** из **NHOBSuperiorToSubordinate**) должен быть полностью представлен в изменении операционного связывания.

25.1.6 Параметры завершения

Ни одна из ролей не предоставляет параметр завершения при завершении NHOB.

25.1.7 Идентификация типа

Неспецифическое иерархическое операционное связывание идентифицируется идентификатором объекта, который присваивается при определении информационного объекта **nonSpecificHierarchicalOperationalBinding OPERATIONAL-BINDING** согласно п. 25.2.

25.2 Определение класса информационного объекта операционного связывания

Данный подраздел определяет тип неспецифического иерархического операционного связывания с использованием шаблона класса информационного объекта **OPERATIONAL-BINDING**, который определяется в Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2.

```
nonSpecificHierarchicalOperationalBinding OPERATIONAL-BINDING::= {
        AGREEMENT       NonSpecificHierarchicalAgreement
        APPLICATION CONTEXTS {
                        { directorySystemAC } }
        ASYMMETRIC
            ROLE-A {            -- вышестоящий DSA
                ESTABLISHMENT-PARAMETER     NHOBSuperiorToSubordinate
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      NHOBSuperiorToSubordinate
                TERMINATION-INITIATOR       TRUE }
            ROLE-B {            -- подчиненный DSA
                ESTABLISHMENT-INITIATOR     TRUE
                ESTABLISHMENT-PARAMETER     NHOBSubordinateToSuperior
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      NHOBSubordinateToSuperior
                TERMINATION-INITIATOR       TRUE }
        ID                  id-op-binding-non-specific-hierarchical }
```

25.3 Процедуры DSA для управления неспецифическим иерархическим операционным связыванием

В описываемых ниже процедурах, как и в процедурах, определенных в п. 24.3, новый DSE или отметка, созданная DSA, должны помещаться в надежное хранилище.

Для обоих процедур образования и изменения, которые описываются ниже, выполняющий роль отвечающего DSA (т. е. не тот, который инициировал образование или изменение) может предоставить DSA, который выполняет роль инициатора, информацию (например, операционные атрибуты), которая по той или иной причине не является допустимой. В таких случаях инициирующий DSA может завершить операционное связывание.

25.3.1 Процедура образования

Только подчиненный DSA может инициировать иерархическое операционное связывание. Причиной этого может являться желание администратора соединить одно или более поддеревьев для содержащихся в DSA статей с определенной точкой в глобальном дереве DIT. В этом случае подчиненный DSA должен установить NHOB в соответствии со следующей процедурой:

1) Подчиненный DSA либо имеет DSE типа **cp** в качестве части существующего контекста именования, либо он создает новый. Он отмечает DSE как *being added* (*добавлен*), генерирует уникальный идентификатор **bindingID** и сохраняет его вместе с DSE контекстного префикса.
2) Подчиненный DSA посылает вышестоящему DSA операцию образовать операционное связывание, которая содержит следующие параметры:
   a) **bindingType** установлен как **nonSpecificHierarchicalOperationalBindingID**;
   b) параметр образования **NHOBSubordinateToSuperior** установлен соответствующим образом;
   c) **NonSpecificHierarchicalAgreement** с компонентом **immediateSuperior** установлен как выделенной имя непосредственного вышестоящего для новой статьи;

d) параметры **bindingID**, **myAccessPoint** и **valid** установлены соответствующим образом.

3) Вышестоящий DSA проверяет, что он является мастером для непосредственного вышестоящего для статьи нового контекстного префикса или возвращает ошибку **operationalBindingError** с проблемой **roleAssignment**.

4) Вышестоящий DSA добавляет DSE типа **nssr** (и информацию атрибута **nonSpecificKnowledge**) к DSE непосредственного вышестоящего для новой записи, сохраняет вместе с ней значение **bindingID** и возвращает результат.

5) Если вышестоящий DSA получает ошибку, то он удаляет DSE нового контекстного префикса и также его отметку. Определение дальнейшей обработки информации для статьи, от которой был получен DSE контекстного префикса, является локальным вопросом.

   Если вышестоящий DSA получает результат, то он добавляет соответствующим образом необходимые DSE типа **glue**, **subentry**, **admPoint**, **rhob** и **immSupr**, которые должны представлять **contextPrefixInfo**; и, в зависимости от обстоятельств, DSE типа **rhob** и **entry**, которые должны представлять **immediateSuperiorInfo**. У DSE контекстного префикса удаляется отметка.

   Если возникает какая-то неисправность (например, неисправность коммуникаций или конечной системы), то подчиненный DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего выполнения образования иерархического операционного связывания, для которого он является инициатором.

25.3.2 Процедура изменения

Если вышестоящий DSA обнаруживает какие-то изменения в информации **NHOBSuperiorToSubordinate**, которую он предоставил подчиненному DSA в рамках неспецифического иерархического операционного связывания, то он должен передать измененную информацию подчиненному DSA. Если NHOB было установлено согласно процедурам п. 25.3.1, то оно должно быть изменено в соответствии с процедурами, определенным для изменения иерархического операционного связывания согласно п. 24.3.2.1 (при этом **NHOBSuperiorToSubordinate** заменяет **SuperiorToSubordinateModification**).

Аналогично, если подчиненный DSA обнаруживает какие-то изменения в информации **NHOBSubordinateToSuperior**, которая была предоставлена вышестоящему DSA, то он должен передать эти изменения вышестоящему DSA. Если NHOB было установлено с использованием процедур из п. 25.3.1, тогда они должны быть изменены в соответствии с процедурой, определенной в п. 24.3.2.2 для изменения иерархического операционного связывания (при этом **NHOBSubordinateToSuperior** заменяет **SubordinateToSuperior**).

25.3.3 Процедура завершения

Для завершения NHOB, которое было установлено с использованием процедур из п. 25.3.1, определяются следующие процедуры.

25.3.3.1 Завершение, инициированное вышестоящим DSA

Завершение иерархического операционного связывания инициируется вышестоящим DSA только в результате административного вмешательства. Необходимо придерживаться следующей процедуры:

1) Вышестоящий DSA отмечает как *being deleted* (*удалено*) соответствующее подчиненному DSA значение в атрибуте **nonSpecificKnowledge**, который хранится в DSE непосредственно вышестоящей записи.

2) Вышестоящий DSA посылает операцию завершить операционное связывание для NHOB, которое установлено с подчиненным DSA. При этом вышестоящий опускает компонент **version** из **bindingID**.

3) Когда подчиненный DSA получает завершить операционное связывание, то он удаляет всю информацию относительно NHOB и посылает результат, если только не является неизвестным компонент **identifier** из **bindingID**. В этом случае возвращается ошибка **operationalBindingError** с проблемой **invalidID**. То, как обрабатывается информация статьи, связанной с подчиненным контекстом именования, является локальным вопросом.

4) Если вышестоящий DSA получает результат или ошибку **operationalBindingError** с проблемой **invalidID**, он должен удалить значение атрибута **nonSpecificKnowledge**, который отмечен как *being deleted* (*удален*) и который представляет информацию точки доступа, связанной с NHOB, и также удаляет всю информацию относительно операционного связывания. Если это было последнее значение атрибута **nonSpecificKnowledge**, он удаляет из DSE атрибут **nonSpecificKnowledge** и DSE типа **nssr**.

   Если возникает какая-то неисправность (например, неисправность коммуникаций или конечной системы), то вышестоящий DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего выполнения завершения иерархического операционного связывания, для которого он является инициатором.

25.3.3.2 Завершение, инициированное подчиненным DSA

Завершение, инициированное подчиненным DSA, может вызываться операцией удаления записи Remove Entry, когда она удаляет последнюю запись в подчиненном контексте именования, запись контекстного префикса для последнего подчиненного контекста именования, хранящегося в подчиненном DSA, или же может вызываться в результате административного вмешательства. Необходимо придерживаться следующей процедуры:

1) Подчиненный DSA отмечает DSE контекстного префикса для контекста именования как *being deleted* (*удалено*).

2) Подчиненный DSA посылает операцию завершение операционного связывания для иерархического операционного связывания с вышестоящим DSA. Подчиненный опускает компонент **version** в **bindingID**.

3) Когда вышестоящий DSA получает операцию завершения операционного связывания, то он удаляет значение атрибута **nonSpecificKnowledge**, который представляет информацию точки доступа, связанную с NHOB, удаляет всю информацию о иерархическом операционном связывании, удаляет атрибут **nonSpecificKnowledge** и DSE типа **nssr** из DSE, который является непосредственным вышестоящим для подчиненного контекста именования (если удаленное значение являлось последним значением атрибута **nonSpecificKnowledge**) и посылает результат, если только не является неизвестным компонент **identifier** из **bindingID** – в этом случае возвращается ошибка **operationalBindingError** с проблемой **invalidID**.

4) Если подчиненный DSA получает результат или ошибку **operationalBindingError** с проблемой **invalidID**, то он должен удалить всю информацию относительно операционного связывания. То, как обрабатывается информация всех статей, связанных с подчиненным контекстом именования, является локальным вопросом.

   Если возникает неисправность (например, неисправность коммуникаций или конечной системы), то вышестоящий DSA должен повторять этапы, начиная с этапа 2) до тех пор, пока не будут получены результат или ошибка для каждого ожидающего завершения NHOB, для которого он является инициатором.

25.4 Процедуры для операций

Операции, которые могут выполняться в кооперативном состоянии для неспецифического иерархического операционного связывания, это те, которые определены в контексте применения **directorySystemAC**.

Процедуры, которым должны следовать DSA, участвующие в неспецифическом иерархическом операционном связывании, должны соответствовать тем, что определены в пп. 16–22.

25.5 Использование контекстов применения

Для того чтобы образовать, изменить или завершить иерархическое операционное связывание с использованием протокола и процедур данного стандарта Справочника, DSA должен использовать контекст применения **operationalBindingManagementAC**.

Приложение A

ASN.1 для распределенных операций

(Данное Приложение является неотъемлемой частью данной Рекомендации | Международного стандарта)

Данное Приложение включает все типы и определения значений ASN.1, которые содержатся в данной спецификации Справочника, в форме модуля ASN.1 **DistributedOperations**.

```
DistributedOperations {joint-iso-itu-t ds(5) module(1) distributedOperations(3) 5}
DEFINITIONS::=
BEGIN

-- EXPORTS All --
-- Типы и значения, которые определены в данном модуле, экспортируются для использования в других
-- модулях ASN.1, содержащихся в данной спецификации Справочника, и для других приложений, которые
-- используют их для доступа к службам Справочника. Другие приложения могут использовать их в своих
-- собственных целях, однако это не должно ограничивать расширения и изменения, которые необходимы для
– поддержания и улучшения служб Справочника.

IMPORTS


--из Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2

        basicAccessControl, commonProtocolSpecification, directoryAbstractService, enhancedSecurity,
        informationFramework,selectedAttributeTypes, serviceAdministration, upperBounds
            FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        DistinguishedName, Name, RDNSequence
            FROM InformationFramework informationFramework

        MRMapping, SearchRuleId
            FROM ServiceAdministration serviceAdministration

        AuthenticationLevel
            FROM BasicAccessControl basicAccessControl

        OPTIONALLY-PROTECTED{ }
            FROM EnhancedSecurity enhancedSecurity


--из Рек. МСЭ-Т X.511 | ИСО/МЭК 9594-3

        abandon, addEntry, CommonResults, compare, directoryBind, list,
        modifyDN, modifyEntry, read, referral, removeEntry, search, SecurityParameters
            FROM DirectoryAbstractService directoryAbstractService


-- из Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5


        ERROR, id-errcode-dsaReferral, OPERATION
            FROM CommonProtocolSpecification commonProtocolSpecification


-- из Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-6

        DirectoryString{}, PresentationAddress, ProtocolInformation, UniqueIdentifier
            FROM SelectedAttributeTypes selectedAttributeTypes

        ub-domainLocalID, ub-labeledURI
            FROM UpperBounds upperBounds;
```

```
-- параметризованные типы для определения связанных операций --

chained { OPERATION: operation } OPERATION::= {
        ARGUMENT  OPTIONALLY-PROTECTED {
            SET {
                chainedArgument ChainingArguments,
                argument        [0]  operation.&ArgumentType } }
        RESULT    OPTIONALLY-PROTECTED {
            SET {
                chainedResult   ChainingResults,
                result          [0]  operation.&ResultType } }
        ERRORS{ operation.&Errors EXCEPT referral | dsaReferral }
        CODE    operation.&operationCode }

-- операции привязывания и развязывания --

dSABind             OPERATION::= directoryBind

--dSAUnbind         OPERATION::= directoryUnbind

-- связанные операции --

chainedRead         OPERATION::= chained { read }

chainedCompare      OPERATION::= chained { compare }

chainedAbandon      OPERATION::= abandon

chainedList         OPERATION::= chained { list }

chainedSearch       OPERATION::= chained { search }

chainedAddEntry     OPERATION::= chained { addEntry }

chainedRemoveEntry  OPERATION::= chained { removeEntry }

chainedModifyEntry  OPERATION::= chained { modifyEntry }

chainedModifyDN OPERATION::= chained { modifyDN }

-- ошибки и параметры --

dsaReferral   ERROR::= {
        PARAMETER OPTIONALLY-PROTECTED {
            SET {
                reference       [0]  ContinuationReference,
                contextPrefix   [1]  DistinguishedName OPTIONAL,
                COMPONENTS OF CommonResults } }
        CODE        id-errcode-dsaReferral }

-- общие аргументы и результаты --

ChainingArguments::= SET {
        originator              [0]  DistinguishedName OPTIONAL,
        targetObject            [1]  DistinguishedName OPTIONAL,
        operationProgress       [2]  OperationProgress
                                     DEFAULT { nameResolutionPhase notStarted },
        traceInformation        [3]  TraceInformation,
        aliasDereferenced       [4]  BOOLEAN DEFAULT FALSE,
        aliasedRDNs             [5]  INTEGER OPTIONAL,
                            -- присутствует только в системах первого издания
        returnCrossRefs         [6]  BOOLEAN DEFAULT FALSE,
        referenceType           [7]  ReferenceType DEFAULT superior,
        info                    [8]  DomainInfo OPTIONAL,
        timeLimit               [9]  Time OPTIONAL,
        securityParameters      [10] SecurityParameters DEFAULT { },
        entryOnly               [11] BOOLEAN DEFAULT FALSE,
        uniqueIdentifier        [12] UniqueIdentifier OPTIONAL,
        authenticationLevel     [13] AuthenticationLevel OPTIONAL,
        exclusions              [14] Exclusions OPTIONAL,
        excludeShadows          [15] BOOLEAN DEFAULT FALSE,
```

```
        nameResolveOnMaster     [16] BOOLEAN DEFAULT FALSE,
        operationIdentifier     [17] INTEGER OPTIONAL,
        searchRuleId            [18] SearchRuleId OPTIONAL,
        chainedRelaxation       [19] MRMapping    OPTIONAL,
        relatedEntry            [20] INTEGER OPTIONAL,
        dspPaging               [21] BOOLEAN DEFAULT FALSE,
        nonDapPdu               [22] ENUMERATED { ldap (0) } OPTIONAL,
        streamedResults         [23] INTEGER OPTIONAL,
        excludeWriteableCopies  [24] BOOLEAN DEFAULT FALSE }

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

DomainInfo   ::=  ABSTRACT-SYNTAX.&Type

ChainingResults::= SET {
        info                [     0]   DomainInfo OPTIONAL,
        crossReferences     [1]  SEQUENCE SIZE (1..MAX) OF CrossReference OPTIONAL,
        securityParameters  [2]  SecurityParameters DEFAULT { },
        alreadySearched     [3]  Exclusions OPTIONAL }

CrossReference::= SET {
        contextPrefix       [0]  DistinguishedName,
        accessPoint         [1]  AccessPointInformation }

OperationProgress::= SET {
        nameResolutionPhase     [0]  ENUMERATED {
            notStarted  (1),
            proceeding  (2),
            completed   (3) },
        nextRDNToBeResolved     [1]  INTEGER OPTIONAL }

TraceInformation::= SEQUENCE OF TraceItem

TraceItem::= SET {
        dsa                 [0]  Name,
        targetObject        [1]  Name OPTIONAL,
        operationProgress   [2]  OperationProgress }

ReferenceType::= ENUMERATED {
        superior                (1),
        subordinate             (2),
        cross                   (3),
        nonSpecificSubordinate  (4),
        supplier                (5),
        master                  (6),
        immediateSuperior       (7),
        self                    (8),
        ditBridge               (9) }

AccessPoint::= SET {
        ae-title            [0]  Name,
        address             [1]  PresentationAddress,
        protocolInformation [2]  SET SIZE (1..MAX) OF ProtocolInformation OPTIONAL,
        labeledURI          [6]  LabeledURI OPTIONAL }

LabeledURI::= DirectoryString{ub-labeledURI}

MasterOrShadowAccessPoint::= SET {
        COMPONENTS OF       AccessPoint,
        category            [3]  ENUMERATED {
            master          (0),
            shadow          (1) } DEFAULT master,
        chainingRequired    [5]  BOOLEAN DEFAULT FALSE }

MasterAndShadowAccessPoints::= SET SIZE (1..MAX) OF MasterOrShadowAccessPoint
```

```
AccessPointInformation::= SET {
        COMPONENTS OF           MasterOrShadowAccessPoint,
        additionalPoints    [4] MasterAndShadowAccessPoints OPTIONAL }

DitBridgeKnowledge::= SEQUENCE {
        domainLocalID       DirectoryString{ub-domainLocalID} OPTIONAL,
        accessPoints    MasterAndShadowAccessPoints }

Exclusions::= SET SIZE (1..MAX) OF RDNSequence

ContinuationReference::= SET {
        targetObject            [0] Name,
        aliasedRDNs             [1] INTEGER OPTIONAL, -- присутствует только в системах первого издания
        operationProgress       [2] OperationProgress,
        rdnsResolved            [3] INTEGER OPTIONAL,
        referenceType           [4] ReferenceType,
        accessPoints            [5] SET OF AccessPointInformation,
        entryOnly               [6] BOOLEAN DEFAULT FALSE,
        exclusions              [7] Exclusions OPTIONAL,
        returnToDUA             [8] BOOLEAN DEFAULT FALSE,
        nameResolveOnMaster     [9] BOOLEAN DEFAULT FALSE }

END -- DistributedOperations
```

Приложение B

Пример распределенного разрешения имени

(Данное Приложение не является неотъемлемой частью данной Рекомендации | Международного стандарта)

На рис. B.1 показан пример того, как распределенное разрешение имени используется для обработки различных запросов Справочника. Пример использует гипотетическое дерево DIT и соответствующую(ие) конфигурацию(ии) DSA, которая(ые) описывается(ются) в Приложении O (Моделирование знания) Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2 и для удобства воспроизводится здесь.

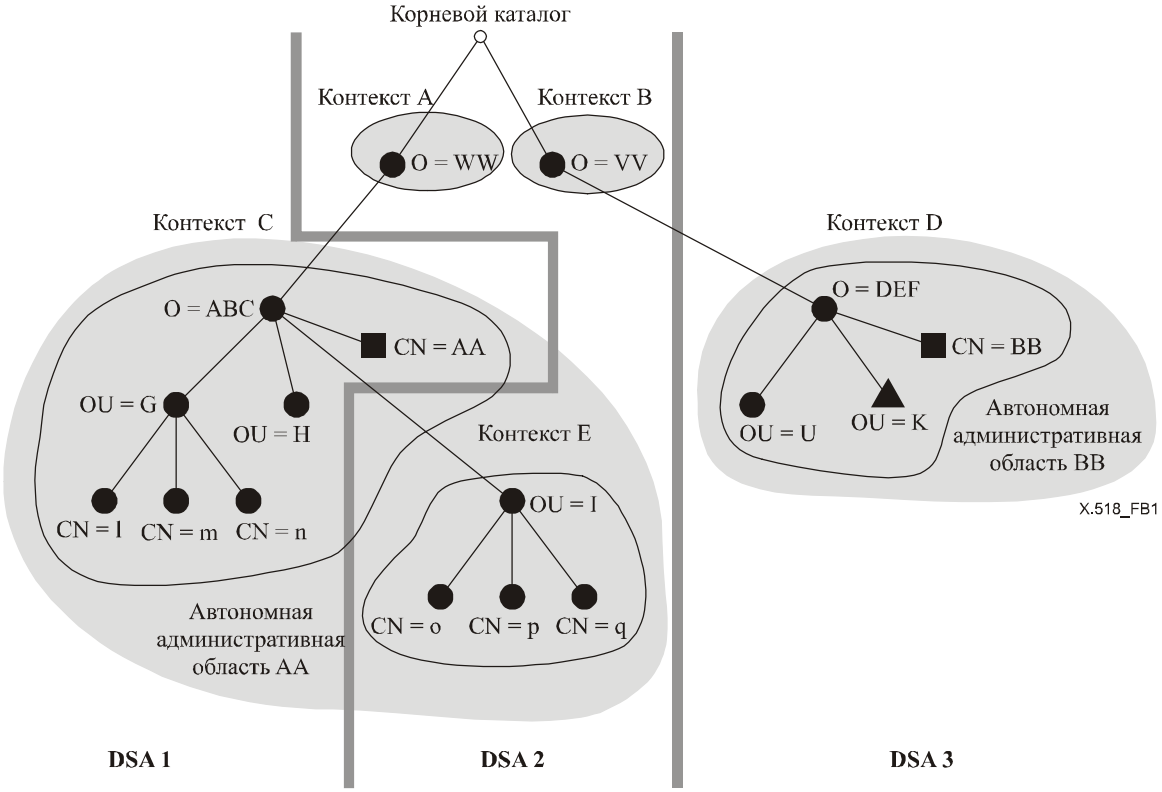


Рисунок B.1 – Гипотетическое дерево DIT и его отображение на три DSA

Предполагая, что используется связанный режим передачи, следующие адресованные DSA 1 запросы будут обрабатываться следующим образом:

1) Запрос с отличительным именем {C = WW, O = ABC, OU = G, CN = l}
   - Разрешение имени успешно определяет каждый RDN в имени цели для DSE, хранящихся в DSA 1 до тех пор, пока будет обнаружен целевой DSE.
2) Запрос с выделенным именем {C = WW, O = JPR}
   - Процедура разрешения имени в DSA 1 обнаруживает сопоставление DSE C = WW и далее неспособна проводить поиск соответствий. В этот момент DSA 1 находит две ссылки, которые могут оказать потенциальную помощь при обработке: одна из них – это ссылка **immSupr** в DSE C = WW, другая является ссылкой **supr** в корневом DSE. В нашем гипотетическом примере обе ссылки указывают на DSA 2. Таким образом, запрос связывается с DSA 2.
   - В DSA 2 процедура **Name Resolution** обнаруживает сопоставление DSE C = WW и далее не может проводить поиск сопоставлений. В этом случае, так как DSE C = WW является **cp** и **entry**, то DSA 2 является главным DSA для данной записи, и далее, так как для C = WW не существует **nssr**, то, следовательно, DSA 2 способен определить, что в Справочнике не содержится такого имени. Возвращается ошибка **nameError** с проблемой **noSuchObject**.

3) Запрос с выделенным именем {C = VV, O = DEF, OU = K}

- Процедура **Name Resolution** в DSA 1 неспособна обнаружить сопоставление для какого-либо DSE. Единственной доступной ссылкой является ссылка **supr** в корневом DSE, которая указывает на DSA 2. Таким образом, запрос связывается с DSA 2.
- В DSA 2 процедура **Name Resolution** обнаруживает сопоставление для DSE C = VV и затем для DSE O = DEF, после чего процедура далее неспособна проводить поиск сопоставлений. Так как обнаруживается, что DSE O = DEF относится к типу **subr**, используется специфическая ссылка на знание, которая указывает на DSA 3, и запрос связывается с DSA 3.
- В DSA 3 процедура **Name Resolution** обнаруживает сопоставление для всего имени целевого объекта и определяет, что обнаруженный DSE относится к типу **alias**. Предполагая, что в данном случае псевдонимы должны быть разыменованы, новое имя конструируется с использованием **aliasedEntryName**, которое содержится в DSE, для которого обнаружено сопоставление. DSA 3 затем для продолжения вновь входит в процедуру **Name Resolution**.

Приложение C

Распределенное применение аутентификации

(Данное Приложение не является неотъемлемой частью данной Рекомендации | Международного стандарта)

C.1 Резюме

Модель безопасности определяется в п. 17 Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2. Ниже приводятся основные отличительные черты данной модели:

a) В DSP поддерживается сильная аутентификация с помощью использования подписи для запроса, результата и ошибок.

b) В DSP поддерживается шифрование запроса, результата и ошибок.

Данное Приложение описывает, как все это может быть реализовано в распределенном Справочнике. При этом используется терминология и обозначения, определенные в Рек. МСЭ-Т X.509 | ИСО/МЭК 9594-8.

C.2 Модель распределенной защиты

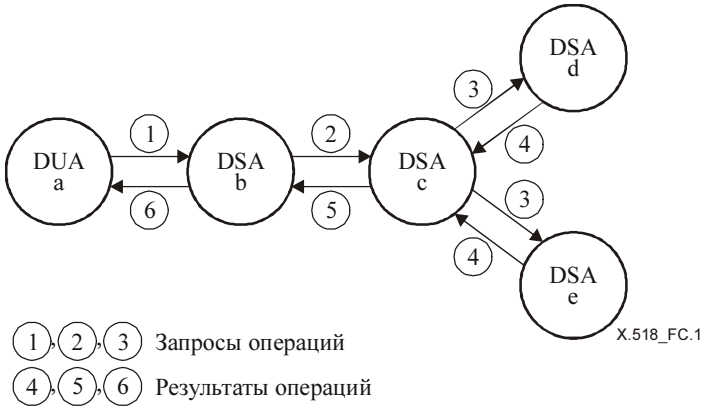


Рисунок C.1 – Распределенная защита

На рис. C.1 показана модель, которая используется при спецификации процедур распределенной защиты. Данная модель определяет последовательность информационных потоков для общего случая операции List (список) или Search (поиск). Считается, что операция начинается из 'a' DUA, при выполнении операции упоминается целевой объект, расположенный в 'c' DSA, и в операции участвуют 'b', 'c', 'd' и 'e' DSA.

'a' DUA первоначально контактирует с любым DSA ('b' DSA), который не содержит целевой объект, но который способен направить с помощью связывания к DSA ('c' DSA), в котором содержится целевой объект. Если все DSA работают в режиме реферала, то модель значительно упрощается и каждый обмен DSA/DSA сводится, в смысле защиты, к взаимодействию между 'a' DUA и 'b' DSA.

C.2.1 Качество защиты

Качество защиты, которое используется в течение срока существования прикладной ассоциации, устанавливается при проведении операции Directory Bind (привязка Справочника). Политика системы определяет уровень защиты, который DUA и DSA обязаны соблюдать. DIRQOP является классом информационного объекта, который может использоваться для спецификации уровня защиты, который должен ассоциироваться с каждой операцией (запроса, результата или ошибки). DUA передает класс информационного объекта **DIRQOP** с помощью **DirectoryBindArgument**, а DSA принимает этот уровень защиты в **DirectoryBindResult**. Качество защиты может использоваться для обеспечения следующих типов защиты: подписанная, зашифрованная или же подписанная и зашифрованная.

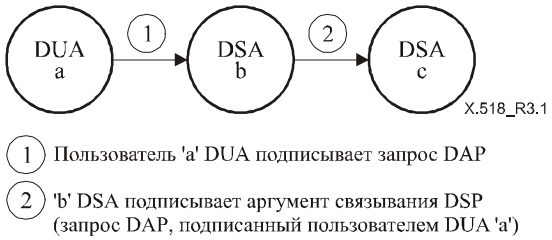
C.3 Подписанные связанные операции

Если поддерживаются операции, снабженные цифровой подписью, то DUA несет ответственность за проверку цифровой подписи, которую возвращают DSA в результате операций List или Search. Если для создания результатов операций List или Search использовалось распределенное окружение, то от DUA требуется способность проверять цифровые подписи для более чем одного DSA. Коррелирование результатов операций List и Search является обязанностью DUA. DSA не должен проводить объединение этих результатов от имени

DUA. В некоторых случаях DUA может получать информацию от различных DSA, которые поддерживают различные уровни аутентификации и цифровых подписей. В этом случае DUA должен самостоятельно решить, стоит ли использовать возвращаемую информацию, если цифровая подпись оказывается недействительной.

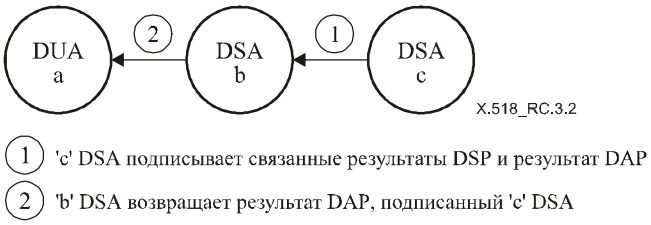
C.3.1 Связанные подписанные аргументы

Если аргумент DAP подписан DUA, то подпись должна поддерживаться на протяжении всего срока существования запроса. Данная подпись может проверяться и использоваться DSA при верификации контроля доступа. Если DSA определяет, что для обработки необходимо связать запрос с другим DSA, то он должен включить подписанный запрос DUA вместе с необходимыми аргументами связывания. Если DSA собирается поддерживать связанные операции DSP (DSA-to-DSA), то рекомендации DSA будут использоваться при подписании для DSP аргументов **ChainingArguments**, также необходимо поддерживать подпись DUA вместе с первоначальным запросом DAP.



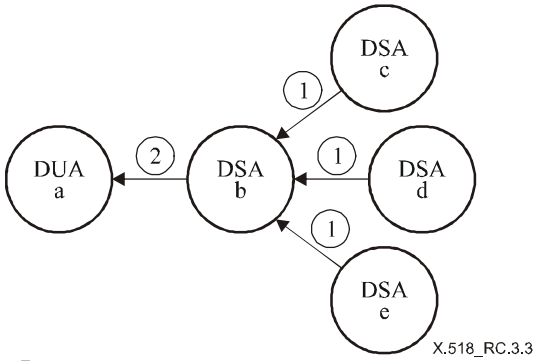
C.3.2 Связанные подписанные результаты

Если пользователь DUA хочет получать подписанные результаты от Справочника, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **SIGNED**. Удаленный DSA должен позволять установить такую конфигурацию, которая позволяет отправлять **ChainingResults** с цифровой подписью. Удаленный DSA может по желанию подписать результат DAP и результаты DSP **ChainingResults**, таким образом поддерживая непрерывную (сквозную) цепь подписей. 'b' DSA отвечает за проверку DSP подписи удаленного DSA, а 'a' DUA будет отвечать за проверку подписи результата DAP (DAP Result Signature) для DSA.



C.3.3 Объединение подписанных результатов для List или Search

Если при создании результатов List и Search использовалось распределенное окружение, то для этого (для объединения подписанных результатов) требуется, чтобы DUA был способен проверять цифровые подписи от нескольких DSA. Корреляция результатов операций List и Search является обязанностью DUA. DSA не должен объединять эти результаты от имени пользователя DUA. В некоторых случаях DUA может получить информацию от различных DSA, каждый из которых поддерживает различные уровни аутентификации и цифровых подписей. Далее DUA должен принять решение о том, следует ли использовать возвращенную информацию в том случае, если цифровая подпись является недействительной.



1 'c', 'd', 'e' DSA подписывают связанный результат DSP (DAP подписан 'c', 'd', 'e' DSA)

2 'b' DSA возвращает частичный результат DAP, подписанный 'c', 'd' и 'e' DSA, 'b' DSA не объединяет результаты DAP

ПРИМЕЧАНИЕ. – DSP протокол DSA-to-DSA также может быть подписан, зашифрован или подписан и зашифрован.

C.3.4 Запрос со множественным связыванием

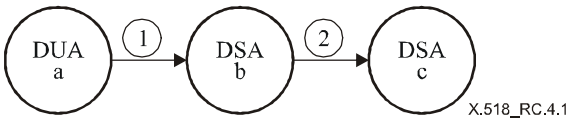
Если DSA определяет, что запрос DAP должен быть связан с несколькими другими DSA, то он может выполнить множественное связывание параллельно или последовательно. Описываются два режима декомпозиции: неспецифическая подчиненная операционная ссылка (NSSR) или декомпозиция запроса. Если выбирается декомпозиция NSSR, то DSA отправляет идентичный запрос другим идентифицированным DSA. При декомпозиции запроса DSA посылает (возможно различные) последовательные запросы каждому из имеющихся DSA.

C.4 Шифрование связанных операций

Если поддерживается шифрование, то необходимо обеспечить эквивалентную защиту между всеми компонентами Справочника. Для того чтобы достичь соглашения по эквивалентности стратегий, требуется установить соответствия, которые выходят за рамки данной спецификации.

C.4.1 Шифрование "пункт-пункт" (DUA->DSA или DSA->DSA) для запроса

Если пользователь DUA хочет зашифровать запрос DAP, то шифрование может осуществляться только на основании метода "пункт-пункт". DUA выполняет шифрование запроса DAP для 'b' DSA, однако пользователь DUA не знает, будет ли данный запрос в конечном счете связан для обработки с удаленным DSA. 'b' DSA расшифровывает запрос и пытается его выполнить. Если 'b' DSA определяет, что запрос должен для обработки быть связан с другим DSA ('c' DSA), то 'b' DSA зашифровывает связанные операции для 'c' DSA. Выбор защиты "пункт-пункт" для запроса DSP и ответов (аргументы и результаты связанной операции) обозначается с помощью **dirqop**, который устанавливается между 'b' DSA и 'c' DSA в операции привязывания DSP.

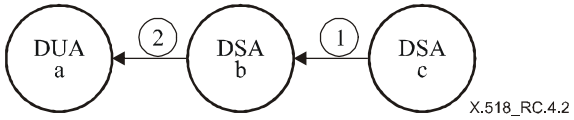


1 Пользователь 'a' DUA шифрует запрос DAP для 'b' DSA

2 'b' DSA шифрует аргумент связанной операции DSP

C.4.2 Шифрование "пункт-пункт" (DUA->DSA или DSA->DSA) для результата

Если пользователь DUA хочет получать от Справочника зашифрованные результаты или ошибки, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**. Если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в **BindArgument** для DAP. Удаленный DSA ('c' DSA) должен позволять задавать конфигурацию для отправки зашифрованных результатов связанной операции. В таком сценарии система 'c' DSA определяет, что может выполнить запрос, затем генерирует результат DAP и результаты связанной операции DSP. Шифрование "пункт-пункт" может выполняться 'c' DSA, который шифрует результаты связанной операции DSP для 'b' DSA. 'b' DSA может расшифровать результаты связанной операции DSP и зашифровать результат DAP для пользователя 'a' DUA. Это обеспечивает для результата шифрование "пункт-пункт". 'a' DUA будет являться ответственным за расшифровку результата DAP своего локального DSA ('b' DSA).

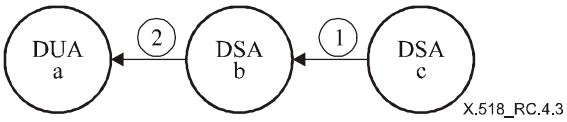


1 'c' DSA шифрует результаты связанной операции DSP

2 'b' DSA шифрует результат DAP для 'a' DUA

C.4.3 Сквозное шифрование результата DAP и шифрование "пункт-пункт" результата связывания DSP

Если пользователь 'a' DUA хочет получать от Справочника зашифрованные результаты или ошибки, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Удаленный 'c' DSA должен позволять задавать конфигурацию для отправки зашифрованных результатов связанной операции. В таком сценарии система 'c' DSA определяет, что может выполнить запрос, затем генерирует сквозное шифрование для результата DAP (для пользователя DUA) и шифрование "пункт-пункт" для результата связанной операции DSP. Шифрование "пункт-пункт" может выполняться 'c' DSA, так как он знает, кто является заданным пользователем 'a' DUA. Шифрование "пункт-пункт" может выполняться для результатов связанной операции DSP с помощью 'c' DSA, который шифрует результаты связанной операции DSP для DSA1. 'b' DSA может расшифровать DSP и ретранслировать зашифрованный результат DAP пользователю 'a' DUA. 'a' DUA будет являться ответственным за расшифровку результата DAP, который он получает от 'c' DSA с помощью 'b' DSA.

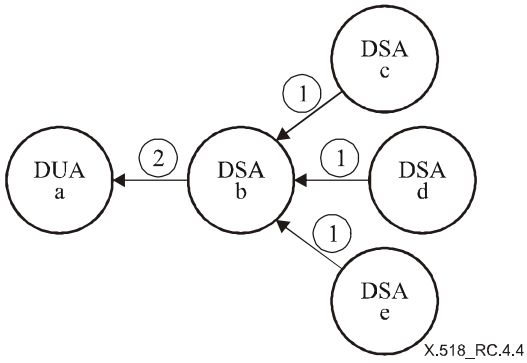


1 'c' DSA шифрует результат связанной операции DSP для 'b' DSA; это включает результат DAP от 'c' DSA, зашифрованный для 'a' DUA

2 'b' DSA возвращает результат DAP, зашифрованный 'c' DSA для 'a' DUA

C.4.4 Объединение результатов List/Search (объединение с повторным шифрованием DSA 1)

Если пользователь 'a' DUA хочет получать от Справочника зашифрованные результаты или ошибки от операций List или Search, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Локальный DSA ('b' DSA) может выбрать множественное связывание для запроса list/search с несколькими другими DSA (параллельное или последовательное). Удаленные DSA ('c' DSA, 'd' и 'e') должны позволять задавать конфигурацию для отправки зашифрованных результатов связанной операции list/search. В этой модели каждый из удаленных DSA ('c', 'd' и 'e') выполняет запрос и создает результаты DAP и зашифрованные результаты связанной операции DSP. Результаты связанной операции, которые были созданы удаленными DSA ('c', 'd' и 'e'), передаются 'b' DSA. 'b' DSA получает каждый из результатов связанной операции, выполняет дешифрование результатов и сравнивает или объединяет эти результаты в один общий результат. Затем 'b' DSA зашифровывает эти новые общие результаты операции list/search и отсылает их пользователю 'a' DUA. Шифрование "пункт-пункт" выполняется удаленными DSA, которые шифруют результаты связанной операции DSP для 'b' DSA, и 'b' DSA, который шифрует результат DAP для пользователя 'a' DUA. DUA будет являться ответственным за выполнение дешифрования одного общего результата DAP.



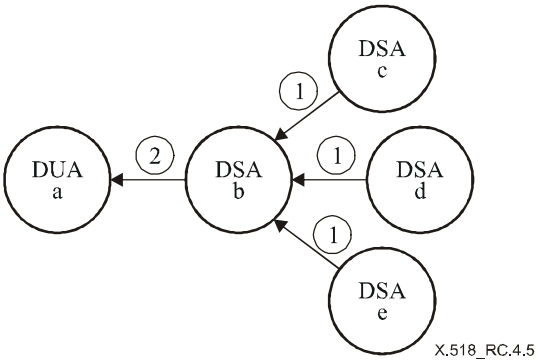
1 'c', 'd', 'e' DSA шифруют результаты связанной операции DSP (вкл. результат DAP)

2 'b' DSA дешифрует результаты связанной операции DSP от 'c' DSA, 'd' DSA и 'e' DSA, затем объединяет результаты DAP и повторно шифрует результат DAP для 'a' DUA

C.4.5 Для результатов List/Search не разрешается объединение

('b' DSA, который выполняет сквозное шифрование для результата операции List/Search DAP, не выполняет объединение)

Если пользователь 'a' DUA хочет получать от Справочника зашифрованные результаты или ошибки от операций List или Search, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Локальный DSA может выбрать множественное связывание для запроса list/search с несколькими другими DSA (параллельное или последовательное). Удаленные DSA ('c', 'd' и 'e') должны позволять задавать конфигурацию для отправки зашифрованных результатов связанной операции list/search. В таком сценарии каждый из удаленных DSA ('c', 'd' и 'e') выполняет запрос и создает зашифрованные результаты DAP (для 'b' DSA) и зашифрованные результаты связанной операции DSP (для 'b' DSA). Результаты связанной операции, которые были созданы удаленными DSA ('c', 'd' и 'e'), передаются 'b' DSA. 'b' DSA получает каждый из результатов связанной операции, выполняет дешифрование результатов и НЕ выполняет никаких сравнений или объединений для этих результатов. 'b' DSA ретранслирует результаты List/Search (которые были зашифрованы 'c', 'd' и 'e') и без каких-либо изменений передает их 'a' DUA. Сквозное шифрование выполняется удаленными DSA, которые шифруют результат DAP операций List/Search для пользователя 'a' DUA, а шифрование "пункт-пункт" выполняется удаленным DSA, который шифрует результаты связанной операции DSP для 'b' DSA. 'a' DUA будет являться ответственным за дешифрование каждого из возвращенных результатов DAP операции List/Search.

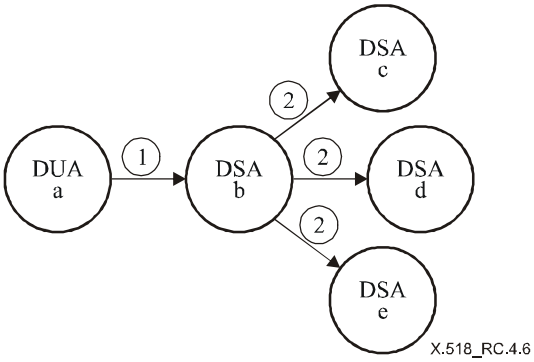


(1) 'c', 'd', 'e' DSA шифруют результаты связанной операции DSP для 'b' DSA; это включает те, которые были зашифрованы для пользователя 'a' DUA

(2) 'b' DSA дешифрует результаты связанной операции DSP от 'c' DSA, 'd' DSA и 'e' DSA, затем транслирует результаты DAP (которые были зашифрованы 'c', 'd' и 'e' для 'a' DUA) без их дешифрования или объединения для 'a' DUA

C.4.6 Множественное связывание запроса DAP с использованием ключа шифрования (net-key)

Если пользователь 'a' DUA хочет получать от Справочника зашифрованные результаты или, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Локальный DSA может выбрать множественное связывание для запроса list/search с несколькими другими DSA (параллельное или последовательное). Локальный DSA ('b' DSA) может быть сконфигурирован для поддержки ключа шифрования или сетевого ключа (net-key). Сетевой ключ является ключом симметричного шифрования и он является общим для всех DSA в данной цепи. С помощью сетевого ключа 'b' DSA должен только один раз выполнить шифрование связанного запроса. Каждый из удаленных DSA знает о сетевом ключе и способен выполнить дешифрование аргумента связанной операции DSP с использованием сетевого ключа. В этом сценарии шифрование "пункт-пункт" может выполняться пользователем DUA, который выполняет шифрование запроса DAP для 'b' DSA, а 'b' DSA может выполнять шифрование "пункт-пункт" для удаленных DSA с помощью сетевого ключа.



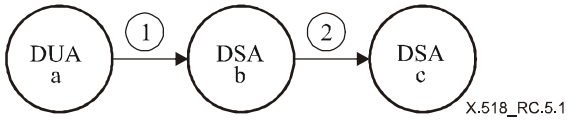
1 'a' DUA шифрует аргумент DAP для 'b' DSA

2 'b' DSA дешифрует запрос и пытается его выполнить; если 'b' DSA не может выполнить запрос, то он использует "net-key" для шифрования запроса связанной операции DSA (включая запрос DAP). Связанный запрос отправляется 'c', 'd' и 'e' DSA

C.5 Подписанные и шифрованные распределенные операции

C.5.1 Сквозные подписи и шифрование "пункт-пункт"

Если пользователь 'a' DUA хочет подписать и зашифровать запрос DAP, то подпись может являться сквозной, а шифрование может выполняться только по принципу "пункт-пункт". 'a' DUA может подписать и зашифровать запрос DAP для 'b' DSA, однако пользователь 'a' DUA не знает о том, будет или нет запрос, в конце концов, связан для обработки с удаленным DSA ('c' DSA). 'b' DSA выполняет дешифрование запроса и проверяет подпись. Затем он пытается выполнить запрос. Если 'b' DSA определяет, что запрос должен связываться для обработки с другим DSA ('c' DSA), то 'b' DSA зашифровывает DSP **ChainingArguments** для 'c' DSA. Первоначальный подписанный запрос DAP также может поддерживаться и передаваться вместе с зашифрованными DSP **ChainingArguments**.

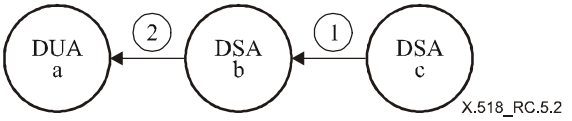


1 Пользователь 'a' DUA подписывает и шифрует запрос DAP для 'b' DSA

2 'b' DSA дешифрует запрос DAP и проверяет подпись; после попытки выполнить запрос локально 'b' DSA определяет, что запрос должен быть связан с 'c' DSA. 'b' DSA отправляет запрос DAP с оригинальной подписью (подпись пользователя 'a' DUA) и создает и шифрует аргумент связывания DSP для 'c' DSA

C.5.2 Сквозная подпись и шифрование для результата DAP, подпись и шифрование "пункт-пункт" для DSP

Если пользователь 'a' DUA хочет получать от Справочника зашифрованные результаты или ошибки, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Удаленный DSA должен позволять задавать конфигурацию для отправки подписанных и зашифрованных связанных операций. В такой модели система 'c' DSA может выполнить запрос, а затем генерирует и выполняет сквозное шифрование для результата DAP (для пользователя 'a' DUA) и шифрование "пункт-пункт" для DSP **ChainingResults**. Подпись и шифрование "пункт-пункт" может выполняться 'c' DSA, так как он знает о том, кто является заданным пользователем 'a' DUA. Подпись и шифрование "пункт-пункт" может выполняться для DSP **ChainingResults** с помощью 'c' DSA, который подписывает и шифрует DSP **ChainingResults** для 'b' DSA. 'b' DSA может дешифровать и проверить подпись 'c' DSA для подписанных DSP **ChainingResults** и ретранслировать подписанный и зашифрованный результат DAP пользователю 'a' DUA. 'a' DUA будет являться ответственным за дешифрование и проверку подписи результата DAP, который он получает от 'c' DSA с помощью 'b' DSA.

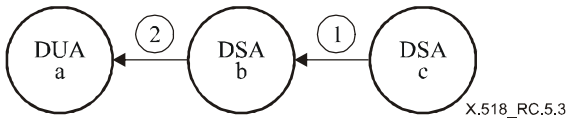


1) 'c' DSA подписывает и шифрует связанный результат DSP для 'b' DSA; это включает результаты DAP, которые подписаны и зашифрованы для пользователя 'a' DUA

2) 'b' DSA дешифрует связанный результат DSP от 'c' DSA и направляет подписанный и зашифрованный результат DAP для 'a' DUA

C.5.3 Сквозная подпись для DAP, шифрование "пункт-пункт" для DSP и результата DAP

Если пользователь 'a' DUA хочет получать от Справочника подписанные и зашифрованные результаты, то поле **SecurityParameters.ProtectionRequest** должно быть установлено как **ENCRYPTED**, а если это поле отсутствует, то поле **SecurityParameters.ProtectionRequest** в аргументах связанной операции должно быть установлено так, чтобы отобразить **DIRQOP** в привязывании DAP. Удаленный DSA ('c' DSA) должен позволять задавать конфигурацию для отправки подписанных и зашифрованных связанных операций. В такой модели система 'c' DSA может выполнить запрос и она подписывает и выполняет шифрование для результата DAP и для DSP **ChainingResults** для 'b' DSA. 'b' DSA может дешифровать и проверить подпись 'c' DSA для DSP **ChainingResults** и повторно зашифровать подписанный (с помощью 'c' DSA) результат DAP для пользователя 'a' DUA. 'a' DUA будет являться ответственным за дешифрование результата DAP, полученного от 'b' DSA, и за проверку подписи результата DAP, который он получает от 'c' DSA с помощью 'b' DSA.



1) 'c' DSA подписывает и шифрует связанный результат DSP для 'b' DSA; это включает результаты DAP

2) 'b' DSA дешифрует связанные результаты DSP от 'c' DSA (и результат DAP, полученный в связанном результате DSP) и отправляет подписанный результат DAP к 'a' DUA

Приложение D

Спецификация типов иерархического и неспецифического иерархического операционного связывания

(Данное Приложение является неотъемлемой частью данной Рекомендации | Международного стандарта)

Данное Приложение включает определения классов информационного объекта ASN.1, который в данной спецификации Справочника представлен в форме ASN.1 модуля **HierarchicalOperationalBindings**.

```
HierarchicalOperationalBindings
        {joint-iso-itu-t ds(5) module(1) hierarchicalOperationalBindings(20) 5}
DEFINITIONS::=
BEGIN

-- EXPORTS All --
-- Типы и значения, которые определены в данном модуле, экспортируются для использования в других
-- модулях ASN.1, содержащихся в данной спецификации Справочника, и для других приложений, которые
-- используют их для доступа к службам Справочника. Другие приложения могут использовать их в своих
-- собственных целях, однако это не должно ограничивать расширения и изменения, которые необходимы для
-- поддержания и улучшения служб Справочника.

IMPORTS


-- из Рек. МСЭ-Т X.501 | ИСО/МЭК 9594-2

        directoryOperationalBindingTypes, directoryOSIProtocols, distributedOperations,
        informationFramework, opBindingManagement
            FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}

        Attribute, DistinguishedName, RelativeDistinguishedName
            FROM InformationFramework informationFramework

        OPERATIONAL-BINDING
            FROM OperationalBindingManagement opBindingManagement


-- из Рек. МСЭ-Т X.518 | ИСО/МЭК 9594-4

        MasterAndShadowAccessPoints
            FROM DistributedOperations distributedOperations


-- из Рек. МСЭ-Т X.519 | ИСО/МЭК 9594-5

        directorySystemAC
            FROM DirectoryOSIProtocols directoryOSIProtocols

        id-op-binding-hierarchical, id-op-binding-non-specific-hierarchical
            FROM DirectoryOperationalBindingTypes directoryOperationalBindingTypes                  ;

-- типы --

HierarchicalAgreement::= SEQUENCE {
        rdn                 [0]  RelativeDistinguishedName,
        immediateSuperior   [1]  DistinguishedName }

SuperiorToSubordinate::= SEQUENCE {
        contextPrefixInfo       [0]  DITcontext,
        entryInfo               [1]  SET SIZE (1..MAX) OF Attribute OPTIONAL,
        immediateSuperiorInfo   [2]  SET SIZE (1..MAX) OF Attribute OPTIONAL }

DITcontext::= SEQUENCE OF Vertex

Vertex::= SEQUENCE {
```

```
        rdn             [0]  RelativeDistinguishedName,
        admPointInfo    [1]  SET SIZE (1..MAX) OF Attribute OPTIONAL,
        subentries      [2]  SET SIZE (1..MAX) OF SubentryInfo OPTIONAL,
        accessPoints    [3]  MasterAndShadowAccessPoints OPTIONAL }

SubentryInfo::= SEQUENCE {
        rdn  [0]  RelativeDistinguishedName,
        info [1]  SET OF Attribute }

SubordinateToSuperior::=  SEQUENCE {
        accessPoints [0]  MasterAndShadowAccessPoints OPTIONAL,
        alias        [1]  BOOLEAN DEFAULT FALSE,
        entryInfo    [2]  SET SIZE (1..MAX) OF Attribute OPTIONAL,
        subentries   [3]  SET SIZE (1..MAX) OF SubentryInfo OPTIONAL }

SuperiorToSubordinateModification::= SuperiorToSubordinate (
        WITH COMPONENTS {..., entryInfo ABSENT})

NonSpecificHierarchicalAgreement::= SEQUENCE {
        immediateSuperior     [1]  DistinguishedName }

NHOBSuperiorToSubordinate::= SuperiorToSubordinate (
        WITH COMPONENTS {..., entryInfo ABSENT})

NHOBSubordinateToSuperior::= SEQUENCE {
        accessPoints [0]  MasterAndShadowAccessPoints OPTIONAL,
        subentries   [3]  SET SIZE (1..MAX) OF SubentryInfo OPTIONAL }

-- информационные объекты операционного связывания --

hierarchicalOperationalBinding OPERATIONAL-BINDING::= {
        AGREEMENT       HierarchicalAgreement
        APPLICATION CONTEXTS {
                        {directorySystemAC} }
        ASYMMETRIC
            ROLE-A {            -- вышестоящий DSA
                ESTABLISHMENT-INITIATOR     TRUE
                ESTABLISHMENT-PARAMETER     SuperiorToSubordinate
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      SuperiorToSubordinateModification
                TERMINATION-INITIATOR       TRUE }
            ROLE-B {            -- подчиненный DSA
                ESTABLISHMENT-INITIATOR     TRUE
                ESTABLISHMENT-PARAMETER     SubordinateToSuperior
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      SubordinateToSuperior
                TERMINATION-INITIATOR       TRUE }
        ID                  id-op-binding-hierarchical }

nonSpecificHierarchicalOperationalBinding OPERATIONAL-BINDING::= {
        AGREEMENT       NonSpecificHierarchicalAgreement
        APPLICATION CONTEXTS {
                        { directorySystemAC } }
        ASYMMETRIC
            ROLE-A {            -- вышестоящий DSA
                ESTABLISHMENT-PARAMETER     NHOBSuperiorToSubordinate
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      NHOBSuperiorToSubordinate
                TERMINATION-INITIATOR       TRUE }
            ROLE-B {            -- подчиненный DSA
                ESTABLISHMENT-INITIATOR     TRUE
                ESTABLISHMENT-PARAMETER     NHOBSubordinateToSuperior
                MODIFICATION-INITIATOR      TRUE
                MODIFICATION-PARAMETER      NHOBSubordinateToSuperior
                TERMINATION-INITIATOR       TRUE }
        ID                  id-op-binding-non-specific-hierarchical }

END -- HierarchicalOperationalBindings
```

Приложение E

Пример поддержания знания

(Данное Приложение не является неотъемлемой частью данной Рекомендации | Международного стандарта)

В данном Приложении с помощью простого примера иллюстрируется поддержание знания, как это определяется в п. 23. На рис. E.1 показаны символы, которые используются для изображения информационных деревьев DSA для пяти DSA.

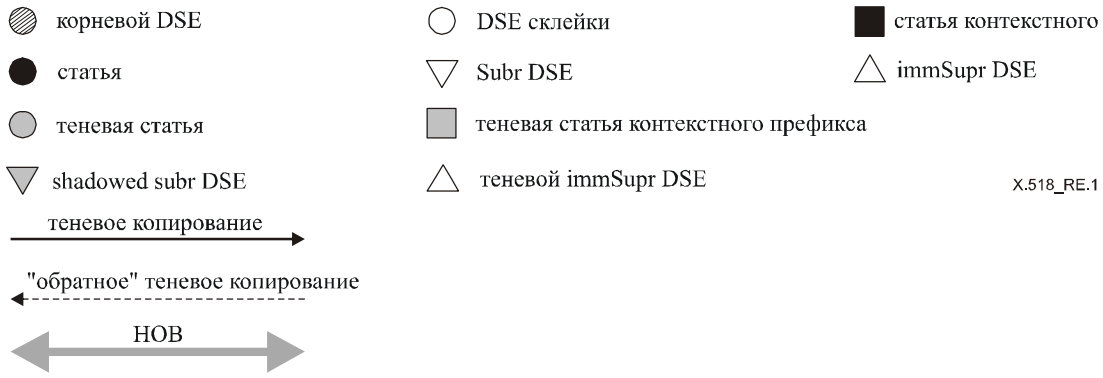


Рисунок E.1 –Символы, которые используются при изображении информационных деревьев DSA

На рис. E.2 DSA 1 является мастером для контекста именования {A}, который состоит из двух статей {A} и {A, B}. DSA 1 содержит подчиненную ссылку для контекста именования {A, B, C}, который поддерживается с помощью НОВ с DSA 3. DSA 1 является теневым поставщиком для DSA 2, который поставляет ему копии информации пользователя контекста именования {A} и подчиненную ссылку на контекст именования {A, B, C}, который идентифицирует точки доступа для DSA 3, DSA 4 и DSA 5, где первый является мастером для подчиненного контекста именования.

DSA 3 является мастером для контекста именования {A, B, C}. В дополнение к хранению единственной статьи {A, B, C} контекста именования, DSA 3 также содержит непосредственную вышестоящую ссылку для контекста именования {A}, которая поддерживается с помощью НОВ с DSA 1. DSA 3 является теневым поставщиком для DSA 4, который поставляет ему копии информации пользователя контекста именования {A, B, C} и непосредственную вышестоящую ссылку на контекст именования {A}, который идентифицирует точки доступа для DSA 1 и DSA 2, где первый является мастером вышестоящего контекста именования. DSA 4 является (вторичным) теневым поставщиком для DSA 5, обеспечивая для него копию информации, которую он получает от DSA 3.

Рис. E.2 иллюстрирует операционные атрибуты DSA, которые используются для представления и поддержания знания.

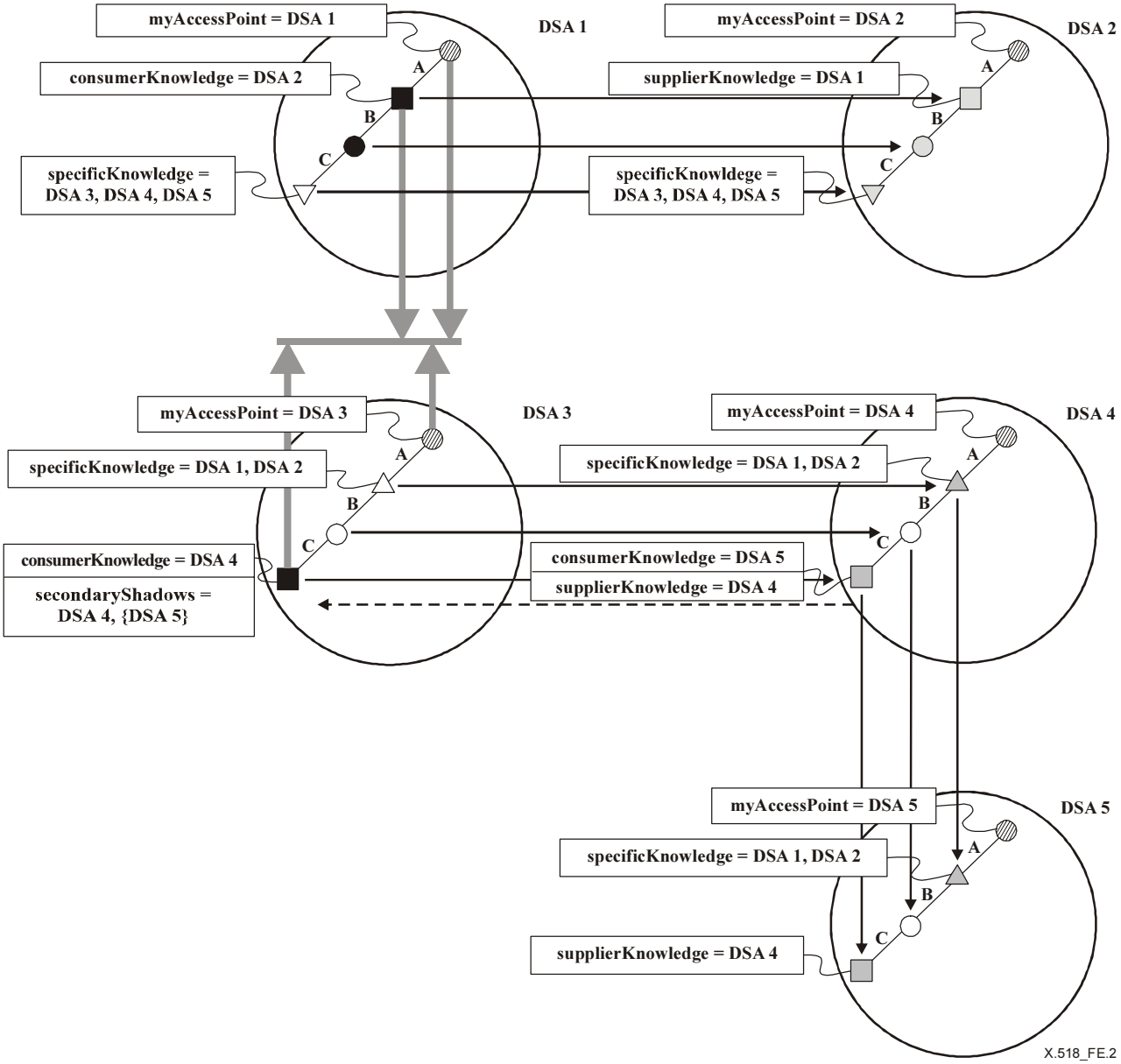


Рисунок E.2 – Пример поддержания знания

DSA 1 использует значение собственного атрибута **myAccessPoint** (связанного с его корневым DSE) и повсеместно используемые значения из своего атрибута **consumerKnowledge** (связанный с контекстным префиксом {A}) для создания значения с типом **MasterAndShadowAccessPoints** для использования во взаимодействии с помощью HOB с DSA 3. В свою очередь DSA 3 использует значение своего атрибута **myAccessPoint** (относящегося к его корневому DSE) и повсеместно используемые значения своего атрибута **consumerKnowledge** и атрибута **secondaryShadows** (оба относятся к контекстному префиксу {A, B, C}) для создания значения с типом **MasterAndShadowAccessPoints** для использования его во взаимодействиях с помощью HOB с DSA 1. Вместе два DSA, используя DOP, поддерживают подчиненную ссылку, которая принадлежит DSA 1, и непосредственную вышестоящую ссылку, которая принадлежит DSA 3. Подчиненная ссылка DSA 1, которая выражается атрибутом **specificKnowledge**, связанным с DSE в {A, B, C}, основывается на значении **MasterAndShadowAccessPoints**, которое оно получает от DSA 3; непосредственной вышестоящей ссылке DSA 3, которая выражается атрибутом **specificKnowledge**, относящемуся к DSE в {A}, таким же образом основывается на значении **MasterAndShadowAccessPoints**, которое оно получает от DSA 1.

DSA 1 и DSA 2 используют собственные значения **myAccessPoint** при взаимодействиях теневого операционного связывания (Shadowing Operational Binding) для поддержания значения **consumerKnowledge** в DSA 1 (который сообщает о точке доступа DSA 2) и **supplierKnowledge** в DSA 2 (который сообщает о точке доступа DSA 1), оба этих атрибута ассоциируются с контекстным префиксом {A}. Вместе два этих DSA с помощью DOP поддерживают ссылку потребителя, которая хранится DSA 1, и ссылку поставщика, которая хранится DSA 2.

DSA 2 получает копию атрибута **specificKnowledge**, связанного с контекстным префиксом {A, B, C} из DSA 1 во взаимодействии DISP с DSA 1. Данное взаимодействие служит для поддержания подчиненной ссылки DSA 2 относительно контекстного префикса {A, B, C}.

DSA 3 и DSA 4 (и аналогичным образом DSA 4 и DSA 5} поддерживают ссылки потребителя и поставщика (соответственно) способом, аналогичным взаимодействию между DSA 1 и DSA 2.

DSA 4 получает копию атрибута **specificKnowledge**, ассоциированного с контекстным префиксом {A4}, от DSA 3 при взаимодействиях DISP с DSA 3. Данное взаимодействие служит для поддержания непосредственной вышестоящей ссылки на контекстный префикс {A} для DSA 4.

DSA 4 сообщает DSA 3 о любых изменениях в собственном атрибуте **myAccessPoint** и атрибуте **consumerKnowledge** (в также атрибуте **secondaryShadows**, который в нашем примере равен нулю) с использованием операции изменения операционного связывания DOP. DSA 4 предоставляет DSA 3 значение **SupplierAndConsumers**, которое содержит только те значения атрибута **consumerKnowledge**, которые идентифицируют те точки доступа DSA, которые имеют повсеместно используемые тени; предоставленные DSA 4 значения атрибута **secondaryShadows**, если только они существуют, согласно проекту все являются повсеместно используемыми. (В этом примере предполагается, что DSA 5 хранит повсеместно используемую копию контекста именования в {A, B, C}.) DSA 3 использует данную информацию для поддержания значения своего атрибута **secondaryShadows**, который ассоциируется с контекстным префиксом {A, B, C}. Как описано выше, этот атрибут используется в взаимодействиях DOP с DSA 1 для поддержания подчиненной ссылки DSA 1 на контекстный префикс {A, B, C}.

DSA 5 поддерживает собственную непосредственную вышестоящую ссылку на контекстный префикс {A} с помощью взаимодействий DISP с DSA 4 способом, аналогичным взаимодействиям между DSA 3 и DSA 4.

Приложение F

Поправки и исправления

(Данное Приложение не является неотъемлемой частью данной Рекомендации | Международного стандарта)

Настоящее издание спецификации Справочника включает следующие проекты поправок к предыдущему изданию, по которым было проведено голосование и которые были утверждены ИСО/МЭК:

- Поправка 1 для расширений в целях поддержки постраничных результатов для DSP;
- Поправка 3 для большей согласованности X.500 и LDAP между собой.

Настоящее издание этой спецификации Справочника включает технические исправления, которые учитывают следующие сообщения о дефектах: 307.

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, 2007 г.