ICS 35.100.70
L 79

中华人民共和国国家标准

GB/T 16264.3—2008/ISO/IEC 9594-3:2005
代替 GB/T 16264.3—1996

信息技术　开放系统互连　目录
第3部分:抽象服务定义

Information technology—Open Systems Interconnection—The Directory—Part 3:Abstract service definition

(ISO/IEC 9594-3:2005 Information technology—Open Systems Interconnection—The Directory:Abstract service definition,IDT)

2008-08-06 发布　　2009-01-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　　次

前　言

GB/T 16264《信息技术　开放系统互连　目录》包括以下10个部分：

——第1部分：概念、模型和服务的概述；

——第2部分：模型；

——第3部分：抽象服务定义；

——第4部分：分布式操作规程；

——第5部分：协议规范；

——第6部分：选定的属性类型；

——第7部分：选定的客体类；

——第8部分：公钥和属性证书框架；

——第9部分：复制(待发布)；

——第10部分：公用目录管理机构的系统管理用法(待发布)。

本部分是GB/T 16264的第3部分。

本部分等同采用ISO/IEC 9594-3:2005《信息技术　开放系统互连　目录　抽象服务定义》，仅有编辑性修改。

本部分代替GB/T 16264.3—1996。

本部分与GB/T 16264.3—1996的差异在于：

——增加13章搜索变元的分析；

——扩展了各章节的功能。

本部分的附录A是规范性附录，附录B和附录C是资料性附录。

本部分由中华人民共和国信息产业部提出。

本部分由全国信息技术标准化技术委员会归口。

本部分起草单位：中国电子技术标准化研究所。

本部分主要起草人：徐冬梅、冯惠、张翠、宋戚阳、胡顺。

本部分于1996年首次发布，本次为第一次修订。

引　言

GB/T 16264 的本部分连同本标准其他部分是为方便信息处理系统之间的互连以提供目录服务而制定的。所有这些系统的集合，连同它们所拥有的目录信息可被视为一个整体，被称为“目录”。目录所拥有的信息，总称为目录信息库(DIB)，典型地被用于方便客体之间的通信、与客体的通信或有关客体的通信等，这些客体如应用实体、个人、终端和分布列表等。

目录在开放系统互连中扮演了重要角色，其目标是，在它们自身的互连标准之外做最少的技术约定的情况下，允许下述各种信息处理系统之间的互连：

——来自不同生产厂商；

——具有不同的管理；

——具有不同的复杂程度，以及

——有不同的年代。

本部分定义了目录为其用户提供的能力。

本部分提供了一些基础框架，在此框架基础上，其他标准化组织和业界论坛可以定义工业配置集。在这些框架中定义为可选的许多特性，可通过配置集的说明，在某种环境下作为必选特性来使用。ISO/IEC 9594 的第 5 版是原有国际标准第 4 版的修订和增强，但不是替代。在系统实现时仍可以声明为符合第 4 版。然而，在某些方面，将不再支持第 4 版(即不再消除一些报告上来的差错)。建议在系统实现时尽快符合第 5 版。

第 5 版详细定义了目录协议的第 1 版和第 2 版。

第 1 版和第 2 版仅定义了协议第 1 版。本版本(第 5 版)中定义的许多服务和协议被设计为可运行在第 1 版下。然而，一些增强的服务和协议，如署名差错，只有包含在操作中的所有的目录条目都协商支持协议第 2 版时才可运行。无论协商的是哪一版，第 5 版中所定义的服务之间的差异和协议之间的差异，除了那些特别分配给第 2 版的外，都可以使用 GB/T 16264.5—2008 中定义的扩展规则调节。

本部分使用术语“第 1 版系统”来指遵循国际标准第 1 版的所有系统，即 ISO/IEC 9594:1990 版本；本部分使用术语“第 2 版系统”来指遵循国际标准第 2 版本的所有系统，即 ISO/IEC 9594:1995 版本；本部分使用术语“第 3 版系统”来指遵循国际标准第 3 版的所有系统，即 ISO/IEC 9594:1998 版本；本部分使用术语“第 4 版系统”来指遵循国际标准第 4 版的所有系统，即 ISO/IEC 9594:2001 版本的第 1 部分到第 10 部分；本部分使用术语“第 5 版系统”来指遵循国际标准第 5 版的所有系统，即 ISO/IEC 9594:2005 版本。

GB/T 16264—1996 是参照 ISO/IEC 9594:1990 而制定的。我国没有制定与国际标准第 2 版、第 3 版、第 4 版对应的国家标准。本部分提到的版本号是指国际标准的版本号。

附录 A 是规范性附录，提供了目录抽象服务的 ASN.1 模块。

附录 B 是资料性附录，提供了用于描述与基本访问控制相关的语义的图表，它适用于目录操作的处理。

附录 C 是资料性附录，给出了条目族使用的例子。

信息技术　开放系统互连　目录
第3部分:抽象服务定义

1　范围

GB/T 16264的本部分按抽象方法定义了目录所提供的外部可视服务。

本部分不规定具体实现或产品。

2　规范性引用文件

下列文件中的条款通过GB/T 16264的本部分的引用而成为本部分的条款。凡是注日期的引用文件,其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本部分,然而,鼓励根据本部分达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件,其最新版本适用于本部分。

GB/T 9387.1—1998　信息技术　开放系统互连　基本参考模型　第1部分:基本模型(idt ISO/IEC 7498-1:1994)

GB/T 16262.1—2006　信息技术　抽象语法记法一(ASN.1)　第1部分:基本记法规范(ISO/IEC 8824-1:2002,IDT)

GB/T 16262.2—2006　信息技术　抽象语法记法一(ASN.1)　第2部分:信息客体规范(ISO/IEC 8824-2:2002,IDT)

GB/T 16262.3—2006　信息技术　抽象语法记法一(ASN.1)　第3部分:约束规范(ISO/IEC 8824-3:2002,IDT)

GB/T 16262.4—2006　信息技术　抽象语法记法一(ASN.1)　第4部分:ASN.1规范的参数化(ISO/IEC 8824-4:2002,IDT)

GB/T 16264.1—2008　信息技术　开放系统互连　目录　第1部分:概念、模型和服务的概述(ISO/IEC 9594-1:2005,IDT)

GB/T 16264.2—2008　信息技术　开放系统互连　目录　第2部分:模型(ISO/IEC 9594-2:2005,IDT)

GB/T 16264.4—2008　信息技术　开放系统互连　目录　第4部分:分布式操作规程(ISO/IEC 9594-4:2005,IDT)

GB/T 16264.5—2008　信息技术　开放系统互连　目录　第5部分:协议规范(ISO/IEC 9594-5:2005,IDT)

GB/T 16264.6—2008　信息技术　开放系统互连　目录　第6部分:选定的属性类型(ISO/IEC 9594-6:2005,IDT)

GB/T 16264.7—2008　信息技术　开放系统互连　目录　第7部分:选定的客体类(ISO/IEC 9594-7:2005,IDT)

ISO/IEC 9594-8:2005　信息技术　开放系统互连　目录:公钥和属性证书框架

ISO/IEC 9594-9:2005　信息技术　开放系统互连　目录:复制

ISO/IEC 9594-10:2005　信息技术　开放系统互连　目录:公用目录管理机构的系统管理用法

3　术语和定义

下列术语和定义适用于GB/T 16264的本部分。

3.1 基本目录定义

下列术语在 GB/T 16264.1—2008 中规定：

a) 目录 directory；

b) 目录信息库 directory information base；

c) (目录)用户 (directory) user。

3.2 目录模型定义

下列术语在 GB/T 16264.2—2008 中规定：

a) 目录系统代理 directory system agent；

b) 目录用户代理 directory user agent。

3.3 目录信息库定义

下列术语在 GB/T 16264.2—2008 中规定：

a) 别名条目 alias entry；

b) 目录信息树 directory information tree；

c) (目录)条目 (directory) entry；

d) 直接上级 immediate superior；

e) 直接上级条目/客体 immediately superior entry/object；

f) 客体 object；

g) 客体类 object class；

h) 客体条目 object entry；

i) 下级 subordinate；

j) 上级 superior；

k) 祖(条目) ancestor；

l) (条目的)家族 family(of entries)；

m) 复合条目 compound entry。

3.4 目录条目定义

下列术语在 GB/T 16264.2—2008 中规定：

a) 属性 attribute；

b) 属性类型 attribute type；

c) 属性值 attribute value；

d) 属性值断言 attribute value assertion；

e) 上下文 context；

f) 上下文类型 context type；

g) 上下文值 context value；

h) 操作属性 operational attribute；

i) 用户属性 user attribute；

j) 匹配规则 matching rule。

3.5 名(称)定义

以下术语在 GB/T 16264.2—2008 中规定：

a) 别名 alias,alias name；

b) 可辨别名 distinguished name；

c) (目录)名(称) (directory) name；

d) 声称名 purported name；

e) 相关可辨别名 relative distinguished name。

3.6 分布式操作定义

以下术语在 GB/T 16264.4—2008 中规定：

a) 绑定 DSA bound DSA；

b) 链接 chaining；

c) 初始执行者 initial performer；

d) 转向推荐 referral。

3.7 抽象服务定义

下列术语和定义适用于本部分。

3.7.1

附加搜索 additional search

指的是从 joinBaseObject 开始的一次搜索，由始发者在 search 请求中规定。

3.7.2

贡献成员 contributing member

复合条目中的一个家族成员，它或者对阅读、搜索或者对修改条目操作做出贡献。

3.7.3

明确未标记的条目 explicitly unmarked entry

依据管理搜索规则引用的控制属性中规定的规范，未包括在 SearchResult 中的一个条目或家族成员。

3.7.4

家族组合 family grouping

出于操作评估目的，将复合属性的成员组合在一起。

3.7.5

过滤器 filter

有关条目的某些属性存在与否或属性值的断言，以便限制搜索范围。

3.7.6

始发者 originator

始发操作的用户。

3.7.7

参与成员 participation member

一个家族成员，或者是一个贡献成员，或是一个家族组合成员，作为整体匹配一个 search 过滤器。

3.7.8

主搜索 primary search

从 baseObject 开始的搜索，按照始发者在 search 请求中规定。

3.7.9

张弛 relaxation

如果接收的太少，为获取更多地匹配条目；或者如果接收的太多，为得到较少的匹配条目，在搜索期间对过滤器行为所做的渐进修改。

3.7.10

服务控制 service controls

作为操作一部分传递的参数，用于约束其性能的不同方面。

3.7.11

束 strand

包括从叶子家族成员一直到祖(条目)路径上的全部成员的家族组合。家族成员将驻于各束中，束

的数量为其下叶家族成员的数量(直接或非直接下级)。

3.7.12

流结果 streamed result

包括在多个响应中的单个操作结果。

4 缩略语

下列缩略语适用于 GB/T 16264 的本部分。

AVA	属性值断言	(Attribute Value Assertion)
DIB	目录信息库	(Directory Information Base)
DIT	目录信息树	(Directory Information Tree)
DMD	目录管理域	(Directory Management Domain)
DSA	目录系统代理	(Directory System Agent)
DUA	目录用户代理	(Directory User Agent)
RDN	相关可辨别名	(Relative Distinguished Name)

5 约定

术语“目录规范(或本目录规范)”指的是 GB/T 16264.3。术语“系列目录规范”指的是GB/T 16264(或者 ISO/IEC 9594)的所有部分。

本目录规范使用术语“第 1 版系统”来指遵循系列目录规范第 1 版的所有系统,即 GB/T 16264—1996 版本。本目录规范使用术语“第 2 版系统”来指遵循系列目录规范第 2 版本的所有系统,即 ISO/IEC 9594:1995 版本。本目录规范使用术语“第 3 版系统”来指遵循系列目录规范第 3 版的所有系统,即 ISO/IEC 9594:1998 版本。本目录规范使用术语“第 4 版系统”来指遵循系列目录规范第 4 版的所有系统,即 ISO/IEC 9594:2001 年版本的第 1 部分到第 10 部分。

本目录规范使用术语“第 5 版系统”来指遵循系列目录规范第 5 版的所有系统,即 GB/T 16264—2008 版本的第 1 部分到第 7 部分以及 ISO/IEC 9594-8:2005、ISO/IEC 9594-9:2005 和 ISO/IEC 9594-10:2005。

本目录规范使用粗体字体来表示 ASN.1 符号。若在常规文本中要表示 ASN.1 的类型和值时,为了区别于常规文本,使用了粗体字表示。为了表示过程的语义而引用过程名时,为了区别于常规文本,使用了粗体字表示。访问控制许可使用斜体字表示。

6 目录服务概述

如 GB/T 16264.2—2008 中所描述,通过 DUA 的访问点提供目录服务,每个访问动作代表一个用户。这些概念如图 1 描述。通过访问点,利用若干目录操作,目录为其用户提供服务。

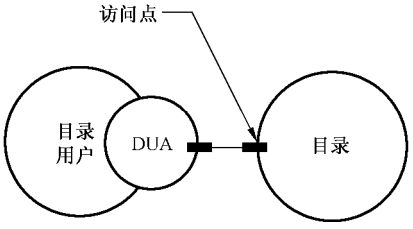

图 1 访问目录

有三种不同类型的目录操作:

a) 目录读操作,它查询单个目录条目;

b) 目录搜索操作,它查询若干潜在的目录条目;以及

c) 目录修改操作。

目录读操作、目录搜索操作和目录修改操作分别在第9章、第10章和第11章中规定。目录操作的一致性在GB/T 16264.5—2008中规定。

7 信息类型和公共规程

7.1 引言

本章标识(在某些情况下定义)后面的目录操作定义中使用的若干信息类型,这里所涉及的信息类型是那些用于多种操作,或在将来用于多种操作的通用信息类型,或者使用这些信息类型定义更复杂的或自包含的信息类型。

目录服务定义中使用的若干信息类型确实在其他地方进行了定义。7.2标识了这些类型,并指示了其定义的来源。7.3至7.10分别标识并定义了一种信息类型。

本章还规定了应用于多数或全部目录操作的若干公共规程元素。

7.2 在其他标准中定义的信息类型

以下信息类型在GB/T 16264.2—2008中规定:

a) Attribute;
b) AttributeType;
c) AttributeValue;
d) AttributeValueAssertion;
e) Context;
f) ContextAssertion;
g) DistinguishedName;
h) Name;
i) OPTIONALLY-PROTECTED;
j) OPTIONALLY-PROTECTED-SEQ;
k) RelativeDistinguishedName。

以下信息类型在GB/T 16264.6—2008中规定:

a) PresentationAddress。

以下信息类型在ISO/IEC 9594-8:2005中规定:

a) Certificate;
b) SIGNED;
c) CertificationPath。

以下信息类型在GB/T 16975.1中规定:

a) InvokeId。

以下信息类型在GB/T 16264.4—2008中规定:

a) OperationProgress;
b) ContinuationReference。

7.3 公共变元

CommonArguments信息可限定目录执行的每个操作的调用。

```
CommonArguments ::= SET {
        serviceControls         [30]    ServiceControls DEFAULT { },
        securityParameters      [29]    SecurityParameters OPTIONAL,
        requestor               [28]    DistinguishedName OPTIONAL,
        operationProgress       [27]    OperationProgress
```

```
                                    DEFAULT { nameResolutionPhase notStarted },
    aliasedRDNs             [26]    INTEGER OPTIONAL,
    criticalExtensions      [25]    BIT STRING OPTIONAL,
    referenceType           [24]    ReferenceType OPTIONAL,
    entryOnly               [23]    BOOLEAN DEFAULT TRUE,
    nameResolveOnMaster     [21]    BOOLEAN DEFAULT FALSE,
    operationContexts       [20]    ContextSelection OPTIONAL,
    familyGrouping          [19]    FamilyGrouping DEFAULT entryOnly }
```

ServiceControls 组件在 7.5 中规定，不存在时表示控制为空集。

SecurityParameters 组件在 7.10 中规定。如果操作变元被请求方签名，那么SecurityParameters 组件应包括在变元中。SecurityParameters 组件缺省时表示为空集。

requestor 可辨别名标识某个抽象操作的始发者，它包含用户与目录建立绑定时使用的用户名。当要对请求签名（见 7.10）时，可以要求这个组件，并且包含发起请求的用户名。

注 1：当用户拥有一个由上下文区分的、可选的可辨别名时，用作requestor 值的名（称）应是所知的主可辨别名。否则，基于requestor 值的鉴别和访问控制可能无法按要求工作。

OperationProgress、referenceType、entryOnly、exclusions 和 nameResolveOnMaster 组件在 GB/T 16264.4—2008 中定义。它们在以下情况下由 DUA 提供：

a) 当按照 DSA 响应先前操作而返回的继续引用进行动作，并且其值由 DUA 从继续引用中复制；或

b) 当 DUA 代表管理 DSA 信息树的管理用户，并且manageDSAIT 选项在服务控制中设置。

aliasedRDNs 组件指示一个 DSA，在先前的操作中，其操作的客体组件通过别名解除引用进行创建，整数值指示客体中 RDN 的数量，它来自没有引用的别名（该值应在以前操作的转向推荐响应中设置）。

注 2：提供本组件是为了与目录第 1 版实现的兼容性。根据目录规范之后版本实现的 DUA（和 DSA）将总是从后续请求的CommonArguments 中省略该参数。这样，如果别名解除引用至进一步别名，目录将不发出差错信号。

operationContexts 组件提供了一组上下文断言，该组上下文断言适用与本操作生成的属性值断言和条目信息选择，否则对于相同的属性类型和上下文类型不包含上下文断言。如果operationContexts 不出现，或不描述某个特定的属性类型或上下文类型，那么 DSA 将使用缺省的上下文断言，如 GB/T 16264.2—2008 的 7.6.1、8.9.2.2 和 12.8 所述。如果选择了allContexts，那么所有属性类型的所有上下文都将是有效的，DSA 提供的各上下文缺省值都将被超越（ContextSelection 在 7.6 中定义）。

对于给定的操作处理，familyGrouping 用于描述应选择哪个家族成员，在 7.3.2 中对它有更为详细的描述。

7.3.1 临界扩展

criticalExtensions 组件提供了一种机制，以列出一组对目录抽象操作的执行来说是临界的扩展。如果扩展操作的始发者希望指示该操作必须和一个或多个扩展一起执行（即没有这些扩展的操作是不能接受的）这种执行是通过设置与该扩展相对应的criticalExtensions 位而进行的。如果目录和目录的某部分不能执行一个临界扩展，它返回一个unavailableCriticalExtension 指示（作为一个serviceError 或一个PartialOutcomeQualifier）。如果该目录不能报告一个非临界的扩展，则它忽略扩展的存在。

本目录规范不建立有关执行 DSA 对其所接收的 PDU 进行解码和处理的次序的规则。收到一个未知临界扩展的 DSA 将返回一个带问题unavailableCriticalExtension 的ServiceError，以发出信号通知操作失败。

目录规范定义了若干扩展。各扩展采用以下形式，即 BIT STRING（比特束）中的额外编号位，或

者集合(SET)或序列(SEQUENCE)中的额外组件,第1版本系统忽视了这一点。每一种扩展被分配了一个可在criticalExtensions中设置成比特数的整数标识符。如果扩展的重要性设为临界,那么DUA将在criticalExtensions中设置相应的位。如果扩展的重要性设为非临界,那么DUA在criticalExtensions中可以或不可以设置相应的位。

扩展、扩展标识符、许可扩展的操作、推荐的临界性、定义扩展的章、相应的LDAP 控制(如果有的话),均示于如表1。

表1 扩展

扩　展	标识符	操　作	临界性	定义(条号)	LDAP控制
subentries	1	所有	非临界	7.5	1.3.6.1.4.1.4203.1.10.1
copyShallDo	2	读、比较、列表、搜索	非临界	7.5	
attribute size limit	3	读、搜索	非临界	7.5	
extraAttributes	4	读、搜索	非临界	7.6	
modifyRightsRequest	5	读	非临界	9.1	
pagedResultsRequest	6	列表、搜索	非临界	10.1	1.2.840.113556.1.4.319
matchedValuesOnly	7	搜索	非临界	10.2	1.2.826.0.1.3344810.2.3
extendedFilter	8	搜索	非临界	10.2	
targetSystem	9	增加条目	临界	11.1	
useAliasOnUpdate	10	增加条目、移除条目、修改条目	临界	11.1	
newSuperior	11	修改DN	临界	11.4	
manageDSAIT	12	所有	临界	7.5、7.13	2.16.840.1.113730.3.4.2
useContexts	13	读、比较、列表、搜索、增加条目、修改条目、修改DN	非临界	7.6、7.8	
partialNameResolution	14	读、搜索	非临界	7.5	
overspecFilter	15	搜索	非临界	10.1.3 f)	
selectionOnModify	16	修改条目	非临界	11.3.2	
安全参数—Response	17	所有	非临界	7.10	
安全参数—Operation code	18	所有	非临界	7.10	
安全参数—Attribute certification path	19	所有	非临界	7.10	
安全参数—Error Protection	20	所有	非临界	7.10	
SPKM Credentials	21	目录绑定	(注3)	8.1.1	
Bind token- Response	22	目录绑定	非临界	8.1.1	

表 1（续）

扩　展	标识符	操　作	临界性	定义(条号)	LDAP 控制
Bind token-Bind Int. Alg, Bind Int Key, Conf Alg and Conf Key Info	23	目录绑定	非临界	8.1.1	
Bind token-DIRQOP(obsolete)	24	目录绑定	非临界	8.1.1	
Service administration	25	读、搜索、修改条目	临界	10.2.2、第 13 章、GB/T 16264.2 第 16 章	
entryCount	26	搜索	非临界	10.1.3	
hierarchySelection	27	搜索	非临界	7.5	
relaxation	28	搜索	非临界	7.8	
familyGrouping	29	比较、搜索、移除条目	非临界 非临界 临界	7.3.2、7.8.3、9.2.2、10.2、11.2.2	
familyReturn	30	读、搜索、修改条目	非临界 非临界 非临界	7.6.4、7.7.1、9.1.3、10.2.3、11.3.3	
dnAttributes	31	搜索	非临界	10.2.2	
friend attributes	32	读、搜索	非临界	7.6、7.8.2	
Abandon of paged results	33	列表、搜索	临界	7.9	
Paged results on the DSP	34	列表、搜索	非临界	7.9	
replaceValues	35	修改条目	临界	11.3.1、11.3.2	

注 1：为首个扩展提供了标识符 1，对应 BIT STRING 的位 1。BIT STRING 的第 0 位没有使用。

注 2：对增加条目、移除条目、修改条目、修改 DN 使用加密的或签名的和加密安全转换或者对任何差错或结果使用保护，要求第 2 版或更高版本的协议。

注 3：SPKM 证书扩展至关重要，除非用在利用第 2 版或更高版本建立的关联中。

7.3.2　家族组合

家族组合允许将复合条目的单个家族成员、若干个家族成员或所有的家族成员结合在一起，以便在操作评估之前做综合考虑。这些语义可以用于以下操作(如下列描述所述)：比较(定义比较属性可能处于的范围)、搜索(定义可能进行过滤的组)、移除条目(定义移除组)。下列 ASN.1 用于选择家族成员。

```
FamilyGrouping ::= ENUMERATED {
        entryOnly           (1),
        compoundEntry       (2),
```

```
    strands                 (3),
    multiStrand             (4) }
```

entryOnly 含义是将在组中对操作选择的特定家族成员进行考虑。这是缺省值，确保向后兼容于目录规范的先前版本。

compoundEntry 含义是将把操作选择的、完整的复合条目看作是一个结合了所有属性的单元。对移除条目操作，只有当规定的客体名(称)是复合条目祖(条目)的客体名(称)时才适用，这将造成全部家族成员被相同操作移除(依据访问控制)。

strands 含义是操作将选择所有与家族成员关联的束。该选项对移除条目操作无效。对搜索操作，认为单个束是用于过滤器目的。如果一个或多个束的复合属性集匹配与过滤器匹配，那么认为复合条目与过滤器匹配。如果基本客体是一个孩子成员，那么只考虑那些通过基本客体的束。对比较操作，条目所属的所有束中所有家族成员的所有属性将会在比较中用到。

multiStrand 只适用于搜索操作，对家族信息限定过滤器的匹配规则。其他操作被忽略。它规定每次只考虑来自复合条目中每个组的一个束，但所有结合在一起考虑。如果基本客体是一个孩子家族成员，那么multiStrand 不适用，在这种情况下，multiStrand 将被忽略，entryOnly 将被替换。

7.4 公共结果

CommonResults 或CommonResultsSeq 信息用于限定目录能执行的各检索操作的结果。另外，它出现在任何返回的差错中。

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

注：CommonResults 和CommonResultsSeq 由相同的组件组成。当被COMPONENT 类型包含在集合类型中时，使用前者，而后者类似地用在序列类型中。

SecurityParameters 组件在 7.10 中规定。如果目录对结果进行签名，那么SecurityParameters 组件将包括在结果中。SecurityParameters 组件不出现将被认为相当于一个空集。

performer 可辨别名用于确定某个特定操作的执行者。当对结果进行签名时可能需要它(见 7.10)，并将持有签名结果的 DSA 的名(称)。

当作为操作目标的客体或基本客体的假设名(称)包括任何已解除引用的别名时，aliasDereferenced 组件将被设为TRUE。

notification 组件将用于限定返回结果和差错 APDU，例如用于提供更加精确的差错信息。标准通告属性在 GB/T 16264.6—2008 的 5.12 中定义。此类通告属性不必储存在目录条目中。

7.5 服务控制

ServiceControls 参数包含指导或限制提供服务的控制信息。

```
ServiceControls ::= SET {
    options                 [0] ServiceControlOptions DEFAULT { },
    priority                [1] INTEGER { low (0),medium (1),high (2) } DEFAULT medium,
    timeLimit               [2] INTEGER OPTIONAL,
```

```
    sizeLimit               [3] INTEGER OPTIONAL,
    scopeOfReferral         [4] INTEGER { dmd(0),country(1) } OPTIONAL,
    attributeSizeLimit      [5] INTEGER OPTIONAL,
    manageDSAITPlaneRef     [6] SEQUENCE {
        dsaName Name,
        agreementID AgreementID } OPTIONAL,
    serviceType             [7] OBJECT IDENTIFIER OPTIONAL,
    userClass               [8] INTEGER OPTIONAL }
ServiceControlOptions ::=BIT STRING {
        preferChaining          (0),
        chainingProhibited      (1),
        localScope              (2),
        dontUseCopy             (3),
        dontDereferenceAliases  (4),
        subentries              (5),
        copyShallDo             (6),
        partialNameResolution   (7),
        manageDSAIT             (8),
        noSubtypeMatch          (9),
        noSubtypeSelection      (10),
        countFamily             (11),
        dontSelectFriends       (12),
        dontMatchFriends        (13),
        allowWriteableCopy      (14) }
```

options 组件包含若干指示，若设置，则每个指示断言所建议的条件。因此：

a) preferChaining 指示优先选择是链接而不是转向推荐提供服务，不强迫目录依从该优先选择。

b) chainingProhibited 指示禁止链接以及其他有关目录的请求分发方法。

c) localScope 指示操作限于本地范围。该选项的定义本身是一个本地问题，例如，在一个单个 DSA 或一个单个 DMD 内。

d) dontUseCopy 指示拷贝的信息(如 GB/T 16264.4—2008 中定义)不会用于提供服务。

e) dontDereferenceAliases 指示不解除引用任何用于标识受操作影响的条目的别名。

注 1：允许引用别名条目本身而不是使用别名的条目，例如为了读别名条目。

f) subentries 指示搜索或列表操作仅用于访问子条目；常规条目变得不可访问，即目录行为如同常规条目不存在。如果不设置该服务控制，那么操作只访问常规条目，子条目变得不可访问。对搜索或列表之外的各操作忽略服务控制。

注 2：即使子条目是不可访问的，仍观察对访问控制、模式和联合属性的子条目影响。

注 3：如果设定该服务控制，那么可以继续将常规条目规定为操作的基本客体。

g) copyShallDo 指示如果目录能够部分地而不是全部地满足对条目拷贝的查询要求，那么它将不链接查询。只有当不设置dontUseCopy 时它才有意义。如果不设置copyShallDo，那么只有当它完整得足以允许操作彻底满足拷贝要求时，目录才使用影像数据。由于在影像拷贝中丢失某些请求的属性，一个查询可能只能部分地满足要求，由于 DSA 不持有它没有的属性值的所有上下文信息，或者由于持有影像数据的 DSA 不支持有关该数据的请求匹配规则，在影像拷贝中会丢失给定属性的某些属性值。如果设置了copyShallDo，并且目录无法彻底满足一个

查询的要求，那么它将在返回的条目信息中设置incompleteEntry。

h) partialNameResolution 指示如果目录只能解析读或搜索操作中的部分声称名，即它将返回一个nameError，那么名(称)包括所有已解析 RDN 的条目将被认为是操作的目标，并且在结果中将partialName 设为TRUE。对读或搜索之外的各操作忽略该服务控制。

注 4：如果设定该服务控制，那么声称名将是一个上下文前缀条目，拒绝对其进行访问，请求方需要访问上级条目，而后将存在上下文前缀条目这一情况间接地泄露给请求方，即使拒绝条目的*DiscloseOnError* 许可。

i) manageDSAIT 指示管理用户已请求操作，因此对 DSA 信息树进行管理。如果在 DSA 有多个复制平面需要管理，并且manageDSAITPlaneRef 服务控制未包括在操作中，那么 DSA 为操作选择一个合适的复制平面。

j) noSubtypeMatch 指示不会尝试进行属性子类型匹配。除了比较和搜索操作，对其他操作将忽略该服务控制。

k) noSubtypeSelection 指示不进行子类型选择。

l) countFamily 指示将把复合条目的每个成员当作一个单独的条目，例如出于大小和管理限制以及张弛控制目的。如果未设置该控制，那么将把复合属性的成员当作一个单个条目。

m) dontSelectFriends 指示条目信息选择中锚属性的规定不自动包括选择中的友人属性。

n) dontMatchFriends 指示过滤器项中锚属性的规定只能满足锚属性值的要求，不能满足友人属性的要求。

o) allowWriteableCopy 指示，在提供查询服务请求中，类型writeableCopy 的 DSE 是可接受的。

注 5：allowWriteableCopy 服务控制不同于copyShallDo，该服务控制用于指示需要一个完整的拷贝，但它不必来自主源，而copyShallDo 用于指示任何拷贝，不论是完整的还是不完整的，都可接受。

如果忽略该组件，那么假设以下内容：对链接没有优选权，但不禁止链接；对操作范围没有限制；许可使用拷贝；将解除引用别名(除非对修改操作，对它不支持别名解除引用)；子条目不可访问；对不能完全满足影像数据要求的操作需做进一步链接。不过，对这些缺省，在服务特定管理区域内可以通过搜索规则进行重写。

以priority (low、medium 或high)优先级提供服务。注意，在目录中这不是一个保证的服务，整体上不进行排队。在低层上使用优先级并不隐含任何关系。

timeLimit 指示服务提供中的最大耗费时间，以秒计。如果约束无法满足，那么报告一个差错。如果忽略该组件，那么不暗指任何时间限制。当在列表或搜索中时间限制超出时，结果是任选一个积累的结果。

注 6：该组件不显示流逝时间中的请求处理时间长度：在处理流逝时间中的请求时可能涉及任何数量的 DSA。

sizeLimit 仅适用于列表和搜索操作。它指示当不返回分页结果时的最大返回条目数。在超出了大小限制的情况下，列表或搜索操作的结果可以是任选一个积累的结果，数量上等于大小限制。将抛弃任何更多的结果。当返回分页结果时，执行分页的 DSA 将忽略 sizeLimit 的值，详见 7.9。

scopeOfReferral 指示 DSA 返回之转向推荐将关联的范围。依据选择的值是dmd 还是country，将只返回选定范围内的其他 DSA 转向推荐。这适用于referral 差错以及list 和search 结果unexplored 参数中的转向推荐。

attributeSizeLimit 指示任何属性的最大大小(即类型及其所有值)，它包括在返回的条目信息中。如果一个属性超出了该限制，那么从返回的条目信息中删去其所有值，并在返回的条目信息中设置incompleteEntry。采用的属性大小为其在持有数据的本地具体语法中的大小，以八位字计。由于所用的不同数据保存方法，限制是不精确的。如果未规定该参数，那么不暗指任何限制。

注 7：作为条目可辨别名一部分返回的属性值不受该限制所限。

priority、timeLimit 和sizeLimit 的某些结合可能产生冲突。例如，短时间限制可能与低优先级产生

冲突;高大小限制可能与低时间限制产生冲突;等等。

manageDSAITPlaneRef 指示,管理用户已请求操作,因此对 DSA 信息树的某个特定复制平面进行管理。如果未设置manageDSAIT 选项,那么忽略manageDSAITPlaneRef 服务控制。平面由dsaName 组件(它是提供 DSA 的名(称))和agreementID 组件(它包含影像协议标识符)确定。

serviceType 服务控制只与search 请求相关,它在一个服务特定管理区域内开始其最初的评估阶段;否则将忽略之。如果提供,那么它增加获得有用通告信息的可能性,当差错表达search 请求时返回通告信息。

userClass 服务控制只与search 请求相关,它在一个服务特定管理区域内开始其最初的评估阶段;否则将忽略之。它确定一个用户类别。它允许请求方规定另一个用户类别,否则将应用目录。如果提供,那么它还会增加获得有用通告信息的可能性,当差错表达 search 请求时返回通告信息。

7.6 条目信息选择

EntryInformationSelection 参数用于指示在检索服务中条目所请求的哪些信息。

```
EntryInformationSelection ::= SET {
    attributes                  CHOICE {
      allUserAttributes             [0]     NULL,
      select                        [1]     SET OF AttributeType
      --empty set implies no attributes are requested--} DEFAULT allUserAttributes: NULL,
    infoTypes                   [2]     INTEGER {
      attributeTypesOnly                (0),
      attributeTypesAndValues           (1) } DEFAULT attributeTypesAndValues,
    extraAttributes                     CHOICE {
      allOperationalAttributes      [3] NULL,
      select                        [4] SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
    contextSelection ContextSelection OPTIONAL,
    returnContexts BOOLEAN DEFAULT FALSE,
    familyReturn FamilyReturn DEFAULT
                                  { memberSelect contributingEntriesOnly } }
ContextSelection ::= CHOICE {
    allContexts                 NULL,
    selectedContexts            SET SIZE (1..MAX) OF TypeAndContextAssertion }
TypeAndContextAssertion ::= SEQUENCE {
    type                        AttributeType,
    contextAssertions           CHOICE {
        preference                      SEQUENCE OF ContextAssertion,
        all                             SET OF ContextAssertion } }
FamilyReturn ::= SEQUENCE {
    memberSelect   ENUMERATED {
        contributingEntriesOnly             (1),
        participatingEntriesOnly            (2),
        compoundEntry                       (3) },
    familySelect SEQUENCE SIZE (1..MAX) OF OBJECT-CLASS.&id OPTIONAL }
```

attributes 组件用于规定有关请求信息的用户和操作属性。

a) 如果选择select 选项,则列出所包含的属性;如果列表为空,则不返回属性;如果属性存在,则

返回所选属性的有关信息；如果没有所选属性存在，则返回attributeError，并指明差错原因为noSuchAttributeOrValue；

b) 如果选择allUserAttributes 选项，则请求条目中的全部属性的有关信息。

如果满足访问权限要求，则只返回属性信息。如果请求不能满足读全部属性的权限要求，则返回securityError，并指明差错原因为insufficientAccessRights。

注 1：注意按照 EntryInformationSelection 组件，访问控制也适用于返回符合条件的属性和值，并可进一步减少返回的信息。

infoTypes 组件规定是请求属性类型和属性值信息（缺省情况下）还是只请求属性类型信息。如果一个类型的某个属性是其他属性的载体，例如，family-information 属性，那么将独立于infoTypes 组件的设置返回值，infoTypes 规范适用于所包含的属性。如果attributes 组件不请求任何属性，那么该组件无意义。

extraAttributes 组件规定一组附加用户和操作属性，这些信息是所请求的。如果选择allOperationalAttributes 选项，那么条目中全部目录操作属性信息被请求。如果选择了select 选项，则列表属性的信息被请求。

注 2：该组件可以用于请求以下信息，例如，当attributes 设为allUserAttributes 时的特定操作属性，或者全部操作属性。如果在attributes 和extraAttributes 中都列出或暗含相同属性，作为请求一次对待。

如果未设置noSubtypeSelection 服务控制选项，那么有关某个特殊属性的请求总被看作是对该属性及其所有子类型的请求（除了由第 1 版本系统处理的请求）。如果设置了noSubtypeSelection 服务控制选项，那么只返回请求的属性，而不返回其子类型。同样，如果dontSelectFriends 服务控制选项没有设置，那么有关某个拥有友人属性的特殊属性的请求，总被看作是对该属性及其所有友人属性的请求。

在响应一个有关属性信息的请求时，目录在对待条目的所有联合属性时就当它们仿佛是条目的实际用户属性，即像其他用户属性一样来选择它们，并合并进返回的条目信息中。有关allUserAttributes 的请求将请求条目的所有联合属性，以及条目的普通属性。如果以下所有各项都为 TRUE，那么属性是条目的一个联合属性：

a) 位于子树规范包括全部条目的子条目中；

b) 可以出现在等同于联合属性类型的collectiveExclusions 属性值条目中；以及

c) 被条目的结构客体类的内容规则所认可。

contextSelection 组件用于规定返回attributes 或extraAttributes 所选之属性中的哪个属性值。只对以下级性值的contextSelection 进行评估，即依据EntryInformationSelection 的其他组件，它们是候选的返回属性。如果它不提供，那么contextSelection 的评估、缺省值的使用将在 7.6.1 到 7.6.3 论述。

如果infoTypes 组件不请求任何属性值，或者attributes 组件不请求任何属性，那么contextSelection 组件没有意义。如果作为应用contextSelection 的结果，没有任何属性值适于返回，那么可以不带任何值地返回属性。

returnContexts 组件用于请求目录带其关联上下文清单地返回属性值。如果该组件不存在或用一个FALSE 值来规定，那么在结果中将不返回任何上下文信息。如果该组件用一个TRUE 值来规定，那么对每个返回的属性值，返回所有的上下文信息。注意，当returnContexts 为TRUE 时，contextSelection 组件对返回什么上下文信息不产生选择性的影响。

如果已标记了一个或多个家族成员，那么familyReturn 组件（如果存在的话）用于确定复合条目中的哪些条目将被返回（见 7.6.4）。

7.6.1 使用 contextSelection 或上下文选择缺省值

contextSelection 组件用于选择attributes 或extraAttributes 所选属性的某些属性值。只能依据候选返回属性值对contextSelection 进行评估，候选属性值依据EntryInformationSelection 的其他组件返回。对每个属性值，管理其属性的任何内容选择都应评估为 TRUE（在 7.6.2 中定义），以便选择该属

性值。

如果出现任何下列组件，那么contextSelection用于管理属性类型：

——ContextSelection用于规定allContexts（在这种情况下，选择所有属性类型的所有属性值）；

——ContextSelection拥有一个selectedContexts，它包括一个TypeAndContextAssertion，其类型等同于属性类型或属性类型的父类型；或者

——ContextSelection拥有一个selectedContexts，它包括一个TypeAndContextAssertion，其类型为id-oa-allAttributeTypes。

如果不提供contextSelection，或它不管理给定的属性类型，那么将应用一个缺省的contextSelection。除了EntryInformationSelection中的contextSelection，还有三个潜在的contextSelection来源：整体上为操作规定的contextSelection；在DIT各分条目中可用的contextSelection；在DSA中本地可用的contextSelection。依据以下优先级来使用它们：

a) 如果contextSelection出现在EntryInformationSelection中，并且它管理给定的属性类型，如上所述，那么将应用它；

b) 如果contextSelection未出现在EntryInformationSelection中，或者它出现了但不管理给定的属性类型。那么如7.3所述已经为操作提供的operationContexts将被应用，如果它存在并且管理给定的属性类型；

c) 如果请求既不是EntryInformationSelection中的contextSelection，也不是操作的operationContexts，或者也不管理给定的属性，那么将应用在控制条目的各上下文断言子条目（如果有的话）中的contextAssertionDefaults属性值，作为selectedContexts（上下文断言子条目在GB/T 16264.2—2008的14.7描述）；

d) 如果上面描述的来源中没有contextSelection来管理给定的属性类型，那么DSA可以应用一个本地定义的缺省contextSelection。这样一个缺省值将典型地反映本地参数，例如语言、DSA的部署位置、当前时间，但对其响应的每个DUA，可以由DSA做不同的剪裁；

e) 如果任何这些来源中都没有任何contextSelection来管理给定的属性类型，那么认为选择了该属性的所有值（即将allContexts当作基本缺省值）。

注：除了管理相同属性类型但作出关于某个不同上下文类型断言的早期contextSelection外，还将应用管理给定属性类型并作出关于某个上下文类型断言的缺省contextSelection，优先级顺序同上所述。

7.6.2 评估 contextSelection

contextSelection将为TRUE（即选择一个给定的属性值），若：

a) 规定了allContexts（这允许一个上下文选择超越任何缺省值，否则如果省略该contextSelection，那么应用缺省值）；或者

b) selectedContexts中的每个TypeAndContextAssertion都为TRUE，如7.6.3所述。否则contextSelection为FALSE。

7.6.3 评估 TypeAndContextAssertion

TypeAndContextAssertion将为TRUE（即选择一个给定的属性值），若：

a) 属性类型不同于TypeAndContextAssertion中的type（也不是其子类型），TypeAndContextAssertion中的type不是id-oa-allAttributeTypes。在这种情况下，TypeAndContextAssertion不适用于特定属性值的属性类型，因此不从选择中去除属性值；或者

b) 对属性值，TypeAndContextAssertion中的contextAssertions为TRUE，如下定义。

注1：OBJECTI DENTIFIER的值id-oa-allAttributeTypes可以用作TypeAndContextAssertion中的type值，以便推动依据属性类型的属性值对contextAssertions进行评估。

contextAssertions表示为一个有关首选上下文的有序序列，或一个有关上下文断言的复合集：

a) 如果规定为all，那么只有当SET中的每个ContextAssertion都为TRUE时，contextAsser-

tions 才为 TRUE，如 GB/T 16264.2—2008 中 8.9.2.4 所定义。

b) 如果规定为preference，那么依据相同属性类型的所有候选属性值依次对 SEQUENCE 中的每个ContextAssertion 进行评估，直至ContextAssertion 评估为 TRUE，如 GB/T 16264.2—2008 中 8.9.2.4 所定义(fallback 标志如果出现，在整个 SEQUENCE 耗尽之前不对它作考虑)。一旦对其中之一的候选属性值ContextAssertion 评估为TRUE，那么将对相同属性类型的每个候选属性值都进行评估，但忽略SEQUENCE 中的后续ContextAssertion。

注 2：preference 提供了一种选择方式，以上下文第一选择、第二选择等形式进行规定(例如，语言＝法语，若没有法语，则语言＝英语)。

否则TypeAndContextAssertion 为 FALSE。

7.6.4 家族返回

如果一个或多个家族成员已标记为贡献成员或参与成员，那么familyReturn 组件用于确定将返回复合条目中的哪些条目。有关如何标记家族成员的规程在 7.13 中做进一步描述。

memberSelect 组件规定在结果中选择哪些条目予以返回：

——contributingEntriesOnly 意味着只返回由操作标记为起作用成员的家族成员。在读或修改条目操作的情况下，它为由 object 操作变元确定的家族成员，对搜索操作，它包括对匹配有影响的家族成员。

——participatingEntriesOnly 意味着只返回由操作标记为参与成员的家族成员。在读或修改条目操作的情况下，同contributingEntriesOnly。

——compoundEntry 意味着将返回复合条目中的每个家族成员，除了那些在搜索操作中可能由管理搜索规则明确未标记的成员。

除了memberSelect 规定的内容，通过规定返回选定家族的所有下级成员，familySelect 组件对memberSelect 组件进行补充。元素的序列并不重要。家族由祖(条目)的家族成员直接下级的结构客体类确定。

如果memberSelect 规定了compoundEntry，那么该组件没有作用。

注：一个管理搜索规则可以修改应返回什么信息(见 GB/T 16264.2—2008 的 16.10)。

7.7 条目信息

7.7.1 条目信息数据类型

EntryInformation 数据类型传递从条目选择的信息。

```
EntryInformation ::= SEQUENCE {
    name                        Name,
    fromEntry                   BOOLEAN DEFAULT TRUE,
    information                 SET SIZE (1..MAX) OF CHOICE {
        attributeType           AttributeType,
        attribute               Attribute } OPTIONAL,
    incompleteEntry     [3]     BOOLEAN DEFAULT FALSE,--不在第 1 版本系统中
    partialName         [4]     BOOLEAN DEFAULT FALSE,--不在第 1 或第 2 版本系统中
    derivedEntry        [5]     BOOLEAN DEFAULT FALSE--不在第 4 版本之前系统中--}
```

Name 参数指的是条目的可辨别名或者条目的别名。无论何时当准许访问控制策略，则返回条目的可辨别名。如果允许对条目的属性而非其可辨别名进行访问，那么目录可以返回一个差错或该条目有效别名的名(称)。

主可辨别名用于Name 参数。这意味着如果形成名(称)的 RDN 包括一个具有多个不同值(由上下文区分)的属性，那么主可辨别值将当作该属性返回 RDN AttributeTypeAndDistinguishedValue 中的值来使用。由于对每个 RD，返回的value 总为主要的不同值，因此将为所有的AttributeTypeAndDis-

tinguishedValue 删去primaryDistinguished。

只有当上下文选择已用于返回的条目信息时，Name 中的 RDN 才包括可选的不同值。可选的可辨别名作为返回 RDN AttributeTypeAndDistinguishedValue 中valuesWithContext 的一部分返回。适用于返回条目信息的上下文选择(见 7.6.1)也适用于可选的不同值，用于确定在valuesWithContext 中使用哪个不同的值。

注 1：内容选择不适用于在Name 中返回的、主要的不同值。

如果已经请求利用结果返回上下文信息，那么上下文信息还将包括在的 Name 中不同值可用的地方(利用 RDN 的valuesWithContext 元素)。当返回可选的不同值时，总为所有的不同值返回上下文信息。

注 2：如果使用别名定位条目，那么该别名应为一个有效的别名。否则，它如何确保别名是有效的将处于这些目录规范的范畴之外。

注 3：当目录的某个特定组件选择返回一个它可用的别名时，建议在以下地方进行选择，即它可能为同一请求方提出的重复请求选择相同的别名，以便提供一致的服务。

fromEntry 参数用于指示信息是取自条目(TRUE)还是取自条目的拷贝(FALSE)。

如果返回条目中的任何属性信息，那么包括information 参数，合适的话，包含一系列attributeTypes 和 attributes。

无论何时当与用户请求相关的返回条目信息不完整时，包括incompleteEntry 参数，并设为TRUE，例如，由于出于访问控制原因删去属性或属性值(以及允许泄露其存在情况)，由于出现不完整的影像信息以及copyShallDo，或者由于超出了attributeSizeLimit。由于已返回别名名(称)而不是可辨别名，因此不设为TRUE。

在考虑partialNameResolution 服务控制之前，目录将在整体上完成操作的名(称)解析阶段(包括在转向推荐后，检查所有的相关知识引用，等等)。如果已耗尽所有的名(称)解析选项，并且已至少解析一个 RDN，那么将包括partialName 参数，如果请求已设置partialNameResolution 服务控制，并且目录无法完成对相关条目所有 RDN 的名(称)解析，那么设为TRUE。当partialName 返回为TRUE 时，它指示返回的信息来自条目，位置在成功解析最后一个 RDN 的地方。

无论何时当返回的条目信息包含联合结果(通过对源自多个目录条目的数据执行联合而获得)，则包括derivedEntry 参数，并设为TRUE。当该参数为TRUE 时，name 的值可以是任何相关条目(条目信息源自这些条目)的名(称)，或者是任何这些条目的别名名(称)。name 的值不得用在后续操作中。如果derivedEntry 参数设为TRUE，并且标记了响应，那么签名为执行联合的 DSA 的签名。

7.7.2 条目信息中的家族信息

当返回来自复合条目的信息时，则依据EntryInformationSelection (管理搜索规则可能对之进行修改)来选择每个待返回成员的属性。当在search 请求中设置了separateFamilyMembers 搜索控制选项时，则每个成员作为一个单独的条目返回。否则，如果返回多个成员，那么条目信息将以如下方式进行包装，即信息看起来像是来自一个单个条目，它可以是祖(条目)或者是一个下级成员(当search 请求的基本客体为下级于祖(条目)的家族成员并且FamilyReturn 尚未选择祖(条目)时，后者是合适的)。其他成员的属性将包装进一个family-information 派生属性中，如下所述。

注 1：依据上述内容，多个家族成员总是包装在一个read 或modifyEntry 结果中。

family-information 导出属性仅用于包装；属性不作为不同的实体存在；它不能直接由entryInformationSelection 选择(将忽略任何有关这方面的尝试)，也不直接受访问控制保护。

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

```
FamilyEntries ::=SEQUENCE {
family-class                    OBJECT-CLASS.&id,--结构客体类值
familyEntries                   SEQUENCE OF FamilyEntry }
FamilyEntry ::=SEQUENCE {
rdn                             RelativeDistinguishedName,
information                     SEQUENCE OF CHOICE {
    attributeType                   AttributeType,
    attribute                       Attribute },
family-info                     EQUENCE SIZE (1..MAX) OF FamilyEntries OPTIONAL }
```

family-information 属性是一个多值属性。如果祖(条目)指定为信息源,那么每个属性值持有来自单个家族的信息。如果作为祖(条目)下级的一个家族成员指定为信息源,那么基于指定成员直接下级成员的结构客体类,信息归类为属性值。

选定的每个家族成员通过类型FamilyEntry 的一个值来描述,它包含:

——选定的属性信息(在适当的地方),作为一个属性类型,或作为一个完整的属性,它取决于EntryInformationSelection 中的infoTypes 值;

注 2:如 7.6 所述,infoTypes 规定只适用于所含的属性,不适用于family-information 属性自身。

——任何以完整的family-information 属性形式出现的、嵌套的FamilyEntries 信息,依据下级条目的结构客体类结合在一起;

——根本不对未选条目进行描述,除非它们是一个或多个所选家族成员的上级。

7.8 过滤器

7.8.1 过滤器参数

Filter 参数提供一个测试,该测试或被特定条目满足或不被满足。过滤器参数根据条目的属性是否存在或某些属性值的断言来表示,当且仅当判断为 TRUE 时,Filter 才能满足。

注:一个过滤器可以是TRUE、FALSE 或UNDEFINED(未定义)。

```
Filter ::=CHOICE {
  item      [0]  FilterItem,
  and       [1]  SET OF Filter,
  or        [2]  SET OF Filter,
  not       [3]  Filter }
FilterItem ::=CHOICE {
  equality          [0]   AttributeValueAssertion,
  substrings        [1]    SEQUENCE {
        type                ATTRIBUTE.&id ({ SupportedAttributes }),
        strings             SEQUENCE OF CHOICE {
          initial                 [0]     ATTRIBUTE.&Type
                                          ({SupportedAttributes}{@substrings.type}),
          any                     [1]     ATTRIBUTE.&Type
                                          ({SupportedAttributes}{@substrings.type}),
          final                   [2]     ATTRIBUTE.&Type
                                          ({SupportedAttributes}{@substrings.type}),
          control                          Attribute } },--用于规定以下项的解释
  greaterOrEqual        [2]     AttributeValueAssertion,
  lessOrEqual           [3]     AttributeValueAssertion,
```

```
    present                 [4]     AttributeType,
    approximateMatch        [5]     AttributeValueAssertion,
    extensibleMatch         [6]     MatchingRuleAssertion,
    contextPresent          [7]     AttributeTypeAssertion }
MatchingRuleAssertion ::=SEQUENCE {
    matchingRule            [1]     SET SIZE (1..MAX) OF MATCHING-RULE.&id,
    type                    [2]     AttributeType OPTIONAL,
    matchValue              [3]     MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
            --matchValue 应是一个类型值,由matchingRule 确定的其中一个
            --MATCHING-RULE 的&AssertionType 字段规定。--} ),
    dnAttributes            [4]     OOLEAN DEFAULT FALSE }
```

一个Filter 是一个FilterItem (见 7.8.2),或者是一个涉及更简单过滤器的表达式,过滤器由逻辑运算符and、or 和not 合成。过滤器的评估结果受张弛策略行为的影响,它可引起用一个匹配规则替代另一个匹配规则,或可提供认为匹配的值。

一个Filter 是一个具有FilterItem 的值(即 TRUE、FALSE 或 UNDEFINED)的FilterItem。

如果设置为空或者如果每个过滤器都为 TRUE,那么一个为一系列过滤器and ("与")运算结果的Filter 值为 TRUE;如果至少一个为 FALSE,那么它为 FALSE;否则它为 UNDEFINED(即至少一个过滤器为 UNDEFINED,并且没有任何一个过滤器为 FALSE)。

如果设置为空或者如果每个过滤器都为 FALSE,那么一个为一系列过滤器or ("或")运算结果的Filter 值为 FALSE;如果至少一个为 TRUE,那么它为 TRUE;否则它为 UNDEFINED(即至少一个过滤器为 UNDEFINED,并且没有任何一个过滤器为 TRUE)。

如果过滤器为 FALSE,那么一个为某个过滤器not ("非")运算结果的Filter 值为 TRUE;如果它为 TRUE,那么Filter 为 TRUE;如果它为 UNDEFINED,那么Filter 为 UNDEFINED。

一个非求反过滤器项定义为一个嵌入于最外Filter 内偶数个(可能为 0 个)not 元素中的过滤器项。因此,一个只含有and 或or 组合中过滤器项的过滤器将只含有非求反项。一个求反过滤器项定义为一个嵌入于最外Filter 内奇数个not 元素中的过滤器项。

7.8.2 过滤器项

FilterItem 是一个有关所测试条目是否出现或者属性值的断言。如果条目包含属性的一个子类型,并且对子类型断言为 TRUE,并且未设置noSubtypeMatch 服务控制选项,或者如果有一个联合条目属性(见 7.6),对其断言为 TRUE,或者如果以下情况,那么有关某个特殊属性类型的断言也是满足要求的。

——不设置dontMatchFriends 服务控制选项;以及

——条目包含一个有关规定属性的友人属性,它有一个兼容于断言的匹配规则;以及

——对友人属性,断言为 TRUE。

每个断言为 TRUE、FALSE 或 UNDEFINED。

每个FilterItem 包括或暗指一个或多个用于确定所考虑之特定属性的AttributeTypes。

有关此类属性值的任何断言只有当评估机制了解AttributeType 时才定义,假设的AttributeValue 符合为该属性类型所定义的属性语法要求,隐含的或指示的匹配规则适用于该属性类型,并且(当使用时)出现的matchValue 符合为指示的匹配规则所定义的语法要求。当不满足这些件时,FilterItem 将评估为逻辑值 UNDEFINED。

注 1: 访问控制限制可能影响对FilterItem 的评估,并可能引起FilterItem 被评估为 UNDEFINED。

另外,如果相关于未出现在属性(其断言正在接受测试)中的属性值和属性类型,那么由这些件定义的断言评估为 UNDEFINED。由这些件定义并且相关于出现之属性值的断言评估 FALSE。

利用为该属性类型定义的匹配规则来评估过滤器项中的属性值断言，合适的话，依据张弛策略的行为，替代该属性类型。依据其定义中的规定对匹配规则断言进行评估。为某个特殊语法定义的匹配规则只能用于生成有关该语法属性或该语法子类型的断言。

注2：张弛策略行为可引起某个特定的匹配规则回复为nullMatch匹配规则（它总评估为TRUE（如果非求反）或FALSE（如果求反）——见GB/T 16264.6—2008的6.7.2）。

一个FilterItem可以为UNDEFINED（如上所述）。否则，当FilterItem断言为：

a) Equality：当且仅当有一个属性值或者equality匹配规则的其中一个子类型应用于该值并且出现的值返回TRUE，它才为TRUE。

b) Substrings：当且仅当有一个属性值或者substring匹配规则的其中一个子类型应用于该值并且在strings中出现的值返回TRUE，它才为TRUE。有关出现的值的语义描述见GB/T 16264.6—2008。

c) greaterOrEqual：当且仅当有一个属性值或者ordering匹配规则的其中一个子类型应用于该值并且出现的值返回FALSE，即有一个属性值大于或等于出现的值，它才为TRUE。

d) lessOrEqual：当且仅当有一个属性值或者equality匹配规则或ordering匹配规则的其中一个子类型应用于该值并且出现的值返回TRUE，即有一个属性值小于或等于出现的值，它才为TRUE。

e) present：当且仅当属性值或者其中一个子类型出现在条目中，它才为TRUE。

f) approximateMatch：当且仅当有一个属性值或者某些本地定义的近似值匹配算法（例如拼写变化、语音匹配等）的其中一个子类型返回TRUE，它才为TRUE。如果一个项匹配等于，那么它也将满足近似匹配。否则在本版目录中没有任何有关近似匹配的特定指南。如果不支持近似匹配，那么该FilterItem应当作是一个equality匹配。

g) extensibleMatch：当且仅当有一个带指示type的属性值或者在matchingRule中规定了匹配规则的其中一个子类型应用于该值并且出现的值matchValue返回TRUE，它才为TRUE。

如果提供了若干匹配规则，那么对这些规则如何结合成一个新规则的方法不做规定（它是一个本地定义算法，反映了组成匹配规则的语义，例如phonetic＋keyword匹配）。

如果省略type，那么对所有兼容该匹配规则的属性类型进行匹配。如果dnAttributes为TRUE，那么除了在评估匹配中所用的那些条目属性之外，还使用条目的可辨别名属性。

如果在filter（而不是在extendedFilter）中请求extensibleMatch，那么将对CommonArguments中criticalExtensions参数中的extendedFilter位进行设置，指示扩展是重要的。

如果实现方案不支持任何在matchingRule子组件中定义的匹配规则，或者没有任何一个匹配规则兼容属性类型，那么若未设置performExactly搜索控制选项，则extensibleMatch过滤器项评估为UNDEFINED。如果设置了performExactly搜索控制选项，那么search请求将被以下拒绝：

——一个带问题unsupportedMatchingUse的serviceError；

——如果不支持所有匹配规则，那么为一个带值id-pr-unsupportedMatchingRule的searchServiceProblem通告属性，否则为一个带值id-pr-unsupportedMatchingUse的searchServiceProblem通告属性；

——一个attributeTypeList通告属性，值同定义了无效匹配规则的属性类型；以及

——一个attributeTypeList通告属性，值同不支持与/或不兼容匹配规则的客体标识符。

注3：对第1版本系统，不允许extensibleMatch。

h) contextPresent：当且仅当该属性类型的AttributeTypeAssertion评估为TRUE，或者若未设置noSubtypeMatch服务控制选项，则其中一个子类型评估为TRUE，它才为TRUE。

如果上下文断言包括在过滤器项的一个属性值断言中，那么只依据那些满足所有给定上下文断言

的值来对过滤器项进行评估，如 GB/T 16264.2—2008 中 8.9.2 所述。如果没有任何上下文断言包括在属性值断言中，那么应用缺省的上下文断言，如 GB/T 16264.2—2008 中 8.9.2.2 所述。

7.8.3 评估带有家族信息的过滤器

在实现过滤器需求中，特定的家族组合工作如下：

entryOnly 意味着只有彻底实现了过滤器需求的那些家族成员才标记为贡献成员和参与成员（有关贡献成员和参与成员的定义见 7.13）。

compoundEntry 意味着整个复合条目形成组，它将满足完整过滤器的要求；在每个复合条目中，它满足过滤器的要求，对匹配起作用的家族成员标记为贡献成员，复合条目的所有成员标记均为参与条目。

strands 意味着过滤器应用于从叶到祖（条目）的每个完整束。如果至少一个束匹配过滤器，那么复合条目匹配过滤器。对匹配起作用的匹配束上的家族成员标记为贡献成员，匹配束上的所有成员标记均为参与条目。

一个束是家族内的一组成员，它们形成从叶到祖（条目）的一路径，由于有许多叶条目，因此会有许多束。

multiStrand 意味着对来自每个家族类别的束的结合是一种出于匹配条目的家族组合。所有的结合在当时都认为是一个。如果至少一个束结合匹配过滤器，那么复合条目匹配过滤器。对匹配起作用的匹配束上的家族成员标记为贡献成员，匹配束结合的所有成员标记均为参与条目。

当且仅当家族成员直接下级于具有相同结构客体类的祖（条目），两个束才能具有相同的家族类别。

当且仅当它出现在至少一个可能的、引起条目匹配分过滤器的束组合中，一个束才能匹配于一个过滤器。以下是必然结果：

——如果祖（条目）完全匹配分过滤器，那么所有的束都匹配。

——同样，如果对某个特定的祖（条目）有三个家族类别，并且两个家族类别满足分过滤器要求，而不考虑第三个家族类别，那么第三个家族类别的所有束都匹配。

只有当基本客体为 DIT 中的祖（条目）（或更高）时，multiStrand 才适用。如果基本客体是一个家族成员，但不是一个祖（条目），那么将忽略multiStrand，并替换entryOnly。

7.9 分页结果

DUA 利用PagedResultsRequest 参数来请求将列表或搜索操作的结果“逐页地”返回给它：它请求 DSA 只返回一个子集--操作结果的--页，特殊地，为下一个 pageSize 的下级或条目，并返回一个 queryReference，它可用于在接下来的查询中请求下一个结果集。

可以由 DSA 来执行分页结果，通过绑定操作，已将 DUA 绑定于其上（绑定的 DSA），或者可以由开始最初评估阶段的 DSA 来执行分页结果（最初的执行方如 GB/T 16264.4—2008 中 15.5.5 详述）。

如果将对结果进行标记，那么将不使用它，除非在 DSA 间达成合作提供分页结果的谅解，这样，执行分页的 DSA 可以从收自其他 DSA 的结果中移除签名，然后它自己对将返回给 DUA 的结果进行标记。建立这种谅解的方式在本目录规范的范围之外。虽然 DUA 可以请求pagedResults，但允许 DSA 忽略结果，并以常规方式返回其结果。

注 1：在配置不是“良好连接”的情况下，结果可能是不可预测的，例如因为影像和使用 NSSR，名（称）解析将确定多个基本客体。

如果请求分页结果并执行了分页，那么如果有的话，分页 DSA 将忽略sizeLimit 服务控制。如果不执行分页，那么将重视sizeLimit 服务控制。一个起作用的 DSA（见 GB/T 16264.4—2008 的 15.5.5）将重视sizeLimit 服务控制。

```
PagedResultsRequest ::=CHOICE {
    newRequest                  SEQUENCE {
      pageSize                      INTEGER,
```

```
    sortKeys                    SEQUENCE SIZE (1..MAX) OF SortKey OPTIONAL,
    reverse                [1]  BOOLEAN DEFAULT FALSE,
    unmerged               [2]  BOOLEAN DEFAULT FALSE,
    pageNumber             [3]  INTEGER OPTIONAL },
  queryReference           OCTET STRING,
  abandonQuery   [0]       OCTET STRING }
SortKey ::= SEQUENCE {
  type                     AttributeType,
  orderingRule     MATCHING-RULE.&id OPTIONAL }
```

对一个新的列表或搜索操作，将PagedResultsRequest 设为newRequest，它由以下参数组成：

a) pageSize 参数用于规定结果中返回的下级或条目的最大数量。DSA 返回的条目数量最多可以达到请求的条目数量，但不超过。如果有的话，将忽略sizeLimit。当封装在familyinformation 派生属性中时，是否包括家族信息不取决于页的大小。

b) sortKeys 参数用于规定一系列属性类型，可选的次序匹配规则用作排序关键字，在返回 DUA 之前对返回的条目进行排序。在列表操作情况下，将通过 RDN 进行排序，但排序要求仅适用于 RDN 中的属性。在搜索操作情况下，排序仅适用于实际提供的属性（作为选择的结果，以可辨别名排序的访问控制作为反馈）。依据序列中第一个SortKey 的type 属性值对各条目进行排序，在多个条目具有相同排序位置的情况下，依据序列中下一个SortKey 的type 属性值进行排序，等等。

   对某个特殊的SortKey，如果它出现，那么 DSA 使用orderingRule 匹配规则，否则如果做了定义，那么使用属性的ordering 匹配规则。如果属性类型是多值的，那么用“最小的”值；如果属性类型从返回的结果中丢失，那么将之看作“大于”所有其他匹配的值。允许一个 DSA 只支持某些排序主要序列（因此，按首字母内部排序的、持有并返回其数据的 DSA 将只能符合一个序列关键序列的要求）。如果它不支持请求的序列，那么它将使用一个缺省的排序序列。不能分隔层次型组，但可以在序列中予以返回，如 GB/T 16264.2—2008 中 10.3 所规定的那样。当进行排序时，返回的层次型组的第一个条目将确定层次型组在排序结果中的位置。

   注 2：一个层次型组可以跨越若干页。

c) 如果reverse 参数为 TRUE，那么 DSA 将以倒序返回排序结果（即从“最大”到“最小”——如果属性类型是多值的，那么使用“最大”值；如果属性类型从返回的结果中丢失，那么将之看作“小于”所有其他匹配的值）。如果reverse 参数为 FALSE，那么 DSA 将以正序返回排序结果。如果没有规定任何sortKeys 参数，那么该参数将被忽略。

d) 如果unmerged 参数为 TRUE，并且负责分页的 DSA 收集来自若干其他 DSA 的结果，那么在返回来自下一个 DSA 的数据之前，它将返回来自某个 DSA 的所有数据（以排序次序）。如果unmerged 参数为 FALSE，那么 DSA 将收集来自所有其他 DSA 的结果，并在返回任何内容之前对合并的数据进行排序。如果没有规定任何sortKeys 参数，那么该参数将被忽略。不论 DSA 是否支持 DSP 分页结果，unmerged 参数的语义都是相同的。

e) 如果pageNumber 参数出现，那么它指示用户想从某个特殊的页开始，而不必从第一页开始。如果未请求排序，那么该参数将被忽略。

对紧接着的请求，即请求分页结果的下一集合，DUA 如之前一样生成列表或搜索请求，当将PagedResultsRequest设为queryReference，该参数的值等同于在前一结果的PartialOutcomeQualifier 中返回的值。DUA 不了解queryReference，它供 DSA 使用，原因是它希望为该查询记录上下文信息。DSA 使用该信息来确定下一个要返回的结果。

通过生成如之前一样的list 或search 请求，通过将PagedResultsRequest 集设为abandonQuery，值

等同于在前一结果的PartialOutcomeQualifier 中返回的queryReference 值，DUA 可以在任何时候指示不需要更多的页了。将不请求或返回更多的页。它的实施依赖于各页何时被清除。

在做出queryReference 或abandonQuery 选择的情况下，新的请求和最初的信息在以下方面将是相同的：

——ListArgument 中SearchArgument 或object 中的baseObject 将呈现的和最初的请求进行匹配；

——pagedResults 的queryReference 子组件等同于在先前结果的PartialOutcomeQualifier 中返回的queryReference 值；

——ServiceControls 数据类型的选项组件将呈现的和最初的请求规定相同的选项；

——operationProgress（如果出现）对呈现的和最初的请求都是一样的。

否则，将返回带问题invalidQueryReference 的serviceError。

注 3：如果在搜索请求之间 DIB 发生了变化，那么 DUA 可能无法见到这些变化的效果。它依赖于执行情况。

注 4：即使 DUA 开始一个新的列表或搜索操作，一个查询引用可以继续保持有效。一个 DUA 请求可以通过若干查询来请求分页结果，而后返回给一个早期的查询，并利用提供给它的查询引用请求下一页结果。DUA 能返回的“活动”查询引用数量是一个本地的 DSA 执行选项，是这些查询引用的生命周期。

注 5：支持abandonQuery 选择仅适用于旧的第 4 版本系统。

注 6：当 DAP 关联终止时，对所有相关分页结果的访问丢失。分页结果只能在最初调用它们的 DAP 相关中进行访问。

7.10 安全参数

SecurityParameters 管理与目录操作有关的各种安全特性的操作。

注 1：安全参数由发送方传送给接收方。当这些参数作为抽象操作的自变元出现时，请求方为发送方，执行方为接收方。结果中，角色相反。

```
SecurityParameters ::= SET {
        certification-path          [0]   CertificationPath          OPTIONAL,
        name                        [1]   istinguishedName           OPTIONAL,
        time                        [2]   Time                       OPTIONAL,
        random                      [3]   BIT STRING                 OPTIONAL,
        target                      [4]   ProtectionRequest          OPTIONAL,
        response                    [5]   BIT STRING                 OPTIONAL,
        operationCode               [6]   Code                       OPTIONAL,
        attributeCertificationPath  [7]   AttributeCertificationPath OPTIONAL,
        errorProtection             [8]   ErrorProtectionRequest     OPTIONAL,
        errorCode                   [9]   Code                       OPTIONAL }
        ProtectionRequest ::= INTEGER { none (0), signed (1) }
Time ::= CHOICE {
        utcTime                     UTCTime,
        generalizedTime             GeneralizedTime }
ErrorProtectionRequest ::= INTEGER { none (0), signed (1) }
```

CertificationPath 组件是一个签名方用户证书的序列，也可有选择地包含一个或多个认证机构（CA）证书的序列（见 ISO/IEC 9594-8:2005 的第 7 章）。用户证书用于绑定签署方的公钥和可辨别名，并可以用来验证对请求变元、响应或差错的签名。如果签署了请求变元、响应或差错，那么该参数将出现，并包含签名方的用户证书。可以出现额外的证书，并可用来确定签名方的用户证书是否有效。如果接收方共用同一认证机构作为签署方，那么不需要额外的证书。如果接收方为确认需要一个鉴别途径，并且未出现一个可接受的参数，那么接收方是否拒绝签名或者尝试确定一个鉴别途径属本地事件。

name 为变元或结果第一个计划中接收方的可辨别名。例如，如果 DUA 产生一个经标记的变元，

那么名(称)为操作提供给它的 DSA 的可辨别名。

注 2：第一个计划的接收方有可选的、由内容区分的可辨别名，name 可以是一个可选的名(称)。不过，如果不使用主可辨别名，那么基于name 值的鉴别和访问控制可能无法如期望的那样开展工作。

time 是计划中的终止时间，针对的是请求、响应或差错的有效性。它与随机数结合使用，使得能够检测重放攻击。

random 值应是一个不同于每个请求、响应或差错的数字。它与时间参数一起使用，以便能够检测重放攻击。如果要求序列完整性，那么随机变元可用于承载一个序列完整性数字，如下所述：

a) 与操作变元一起使用的随机值利用来自以下的预先商定序列(例如，前一个值加 1)获得：
   1) 对绑定中从系统发送的第一个操作，由远程对等系统在绑定操作变元/结果中传送的随机值；以及
   2) 对后续操作，在相同方向的前一个操作中传送的随机值。
b) 与操作结果或差错一起使用的随机值利用来自请求中随机值的预先商定序列(例如，请求变元中的随机数加 1)获得：

target、ProtectionRequest 只可出现于待完成的操作请求中，并指示请求方有关提供给结果的保护等级的优先级。提供了两个保护等级：none(没有请求保护，缺省值)，以及signed(请求目录对结果进行标记)。实际提供给结果的保护等级以结果形式指示，依据目录的限制，它可能等于或小于所请求的等级。

response 用于将任何信息传送回请求的始发者。

operationCode 用于将操作代码安全地绑定于请求变元、结果或差错。

attributeCertificationPath 用于为基于规则的访问控制传送一个安全清晰说明，或属性证书中的其他属性，确认属性证书所需的证书为可选。

errorProtection 只可出现于待完成的操作请求中，并指示请求方有关提供给任何差错的保护等级的优先级。提供了两个保护等级：none(没有请求保护，缺省值)，以及signed(请求目录对结果进行标记)。实际提供给差错的保护等级以差错形式指示，依据目录的限制，它可能等于或小于所请求的等级。

注 3：DUA 可以要求任何安全标签上下文都应利用上下文选择返回一个属性值。

当响应一个操作返回一个差错时，errorCode 用于保护差错代码。

如果Time 的语法选为UTCTime 类型，那么 2 个数字表示的年字段的值将被解释为 4 个数字表示的年值，如下所示：

——如果 2 个数字表示的值为 00～49(包含)，那么值将加上 2000。

——如果 2 个数字表示的值为 90～99(包含)，那么值将加上 1900。

如果商定的版本为 v2 或更高版本，那么将使用GeneralizedTime。当商定结果为 v1 时，GeneralizedTime 的使用可能有碍相互作用，实现方案无法感知是可能选择UTCTime 还是选择GeneralizedTime。这是那些用于规定域的版本的责任，在这些域中将使用目录规范，例如描述概貌组，什么时候可以使用GeneralizedTime。UTCTime 将不用于描述任何超过 2049 年的日期。

7.11 访问控制规程的公共元素

当basic-access-control、rule-based-access-control 或二者都起作用时，本小用于定义所有抽象服务操作公用的规程元素。如果两种机制都起作用，那么其应用次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么来自其他机制的许可将不超越它。在这种情况下，basicaccess-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

7.11.1 基本访问控制规程的公共元素

7.11.1.1 解除引用别名

如果在定位目标客体条目(在抽象服务操作变元中确定)过程中要求解除引用别名，那么为了解除引用别名不需要特定的许可。不过，如果别名解除引用将产生返回一个ContinuationReference(即在

Referral 中),那么应用以下访问控制序列。如果 DSA 将请求链接至另一个 DSA,并从其处收回一个转向推荐,那么若转向推荐中的targetObject 等同于链接请求中的targetObject,则访问控制将适用于转向推荐。也就是说,DSA 将对所有的转向推荐进行监管,不论其是本地产生的还是远程产生的。

a) 别名条目需要取得读许可。如果未赋予许可,那么依据 7.11.1 中所述的规程,操作失败。

b) aliasedEntryName 属性及其所含的单个值需要取得读许可。如果未赋予许可,那么操作失败,并返回一个带问题aliasDereferencingProblem 的nameError。matched 元素将包含别名条目的名(称)。

注:除了上面所述的访问控制,安全策略还可以防止泄漏知识信息,否则它将作为Referral 中的ContinuationReference 予以传送。如果这样一个策略发挥作用,并且如果 DUA 通过规定chainingProhibited 来约束服务,那么目录可能返回一个带问题chainingRequired 的serviceError。否则,将返回一个带问题insufficientAccessRights 或noInformation 的securityError。

7.11.1.2 返回名(称)差错

如果在执行抽象服务操作时不能找到特定的目标客体(别名或条目)——例如,待读条目的名(称)或search 请求中的baseObject,那么将返回一个带问题noSuchObject 的nameError。matched 元素将包含下一个赋予了*DiscloseOnError* 许可的上级条目的名(称),或者 DIT 根的名(称)(即一个空的RDNSequence)。

注:第二个选项可以通过 DSA 获得,它不访问所有的上级条目。

7.11.1.3 不泄露条目是否存在

如果在rule-based-access-control 下拒绝访问,那么*DiscloseOnError* 许可不适用。

如果在执行抽象服务操作时特定的目标客体条目未赋予所需的条目级许可——例如,待读条目,那么操作失败,返回的差错为以下之一:如果条目标条目赋予了*DiscloseOnError* 许可,那么返回一个带问题insufficientAccessRights 或noInformation 的securityError;否则,返回一个带问题noSuchObject 的nameError。matched 元素将包含下一个赋予了*DiscloseOnError* 许可的上级条目的名(称),或者 DIT 根的名(称)(即一个空的RDNSequence)。

注:第二个选项可以通过 DSA 获得,它不访问所有的上级条目。

另外,无论何时当目录检测到一个操作差错(包括转向推荐),它将确保在返回该差错时不否认已命名目标条目及其任何上级的存在。例如,在返回一个带问题timeLimitExceeded 的serviceError 或带问题notAllowedOnNonLeaf 的updateError 之前,目录将确认*DiscloseOnError* 许可赋予了目标条目。如果未赋予,那么接着执行上面段落中所述的规程。

7.11.1.4 返回可辨别名

在比较、列表或搜索操作中,如果是作为解除引用别名的结果,那么object (或baseObject)条目需要取得*ReturnDN* 许可,客体的可辨别名将在操作结果的name 参数中返回(见 9.2.3)。如果未赋予该许可,那么目录将为条目返回一个别名,如 7.7 所述,或者将全部忽略name 参数。

在读或搜索操作中,为了返回其在EntryInformation 中的可辨别名,条目需要取得*ReturnDN* 许可。如果未赋予该许可,那么目录将返回一个别名,如 7.7 所述,或者如果没有任何别名可用,那么操作将失败,产生一个nameError 差错(在读操作情况下),或者从结果中忽略该条目(在搜索操作情况下)。

如果在结果中返回用户提供的别名,那么不得将CommonResults 的aliasDeferenced 标志设为TRUE。

7.11.2 基于规则的访问控制规程的公共元素

7.11.2.1 访问条目(条目级许可)

为了访问一个条目,需要得到至少访问条目中一个属性值的许可。如果未赋予条目级的许可,那么将返回带问题noSuchObject 的nameError。

7.11.2.2 返回条目名(称)

为了返回一个条目的 DN,需要允许访问至少一个条目 RDN 上下文变元的所有属性值(术语定义

为*RDN* 许可)。对条目的任何上级没有任何许可要求。如果未赋予 RDN 许可,那么 DSA 可以选择为已赋予 RDN 许可的属性值返回一个条目有效别名的 DN,或者从操作结果中删去名(称)组件。

注:有关适当别名的选择将在 7.7 的注释中做进一步描述。

7.11.2.3 解除引用别名

为了解除引用别名,需要得到访问aliasedEntryName 属性值的许可。

7.11.2.4 返回名(称)差错(noSuchObject)

带问题noSuchObject 的nameError 的matched 组件将设为下一个上级条目的名(称),其请求方拥有 RDN 许可。如果这样一个条目不适用于产生差错的 DSA,那么将返回 DIT 根的名(称)。

7.11.2.5 访问属性

为了访问一个属性,需要得到至少访问属性中一个值的许可。

7.11.2.6 删除信息

为了删除一个属性值,需要取得访问该值的许可。当删除一个条目或一个属性时,如果至少删除了一个属性值,那么操作将返回一个成功响应,而不管请求删除多少值。

7.11.2.7 调用搜索规则

为了依据搜索操作的变元对搜索规则进行评估,发起搜索操作的请求方需要调用搜索规则许可。为了访问搜索规则属性或包含它的分条目,用户不需要任何其他许可。

7.11.3 家族信息

家族信息当作为任何其他信息,除了 ACI,其ProtectedItem 标记为includeFamily;如果 ACI 适用于祖(条目)或家族成员,那么这将引起下级家族成员受制于同一 ACI。只有当应用于entry 保护项时,IncludeFamily 才有意义。

7.12 管理 DSA 信息树

由 DSA 持有的 DSA 信息树可以利用目录抽象服务进行管理。当管理 DSA 信息树时:

——DSA 中的所有 DSE 通过 DAP 都是可见的,包括根 DSE;

——规定不为用户修改的属性可被修改(虽然如果它不支持请求的修改,DSA 可以利用带问题unwillingToPerform 的serviceError 进行回复);

——知识只是另一个可以读和修改的属性;以及

——DSA 从不链接请求或返回转向推荐或连续引用。

DSE 的可见性和操作属性的检索或修改可以通过常规方式的访问控制进行控制。

DSA 信息树的管理通过使用以下规程的 DUA 实现:

a) DUA 直接绑定于持有 DSA 信息树的 DSA 上,将要对之进行管理;

b) 对每个用于管理 DSA 信息树的操作:

——将设置manageDSAIT 扩展位;

——将设置manageDSAIT 选项;

——如果需要管理特定的复制平面,那么将包括manageDSAITPlaneRef 选项;

目录忽略以下组件:

——CommonArgument 中的operationProgress;

——CommonArgument 中的referenceType;

——CommonArgument 中的entryOnly;

——CommonArgument 中的nameResolveOnMaster;以及

——ServiceControls 中的chainingProhibited。

7.13 条目家族规程

如 7.3.2 所规定,出于操作评估目的,可以将复合条目内的家族成员组合在一起。这种组合只与比较、搜索和移除条目操作有关。如果家族组合是为任何其他操作规定的,那么忽略之。

为了依据entryInformationSelection 的familyReturn 组件确定将返回哪些家族成员，引入了*贡献成员*和*参与成员*两个概念。这些概念只与将返回条目信息的操作相关，即读、搜索和修改条目操作。

如果家族成员对操作评估起积极的作用，那么它标记为贡献成员。如果它是匹配过滤器的家族组合的一部分，并且如果它持有一个或多个匹配于非求反过滤器项的属性，那么家族成员对匹配起作用。如果它持有某个给定类型的属性，并且如果相同类型的求反过滤器项不匹配，那么它也起作用。在读或修改条目操作中，只有操作选择的家族成员（如操作的object 组件所规定）才能被标记为贡献成员和参与成员。在搜索操作中，家族组合针对的是过滤器匹配。如果家族组合匹配过滤器（见 7.8.3），那么所有对匹配起积极作用的成员都将被标记为贡献成员，而组的所有条目都被标记为参与成员。如果所用的过滤器是默认的过滤器（and：{ }），那么家族组合的所有成员都将被标记为参与成员，但不被标记为贡献成员。

当复合条目的家族组合匹配过滤器并且SearchArgument 规定了层次型选择（除了self）时，合适的话，对所选的条目也做标记。如果复合条目的祖（条目）标记为参与成员（也有可能标记为贡献成员），那么将选择不是复合条目的、层次型组的所有引用条目，否则将之排除在外。如果引用的条目是一个复合条目，那么按以下所述对其成员进行标记。以相同的方式对与匹配复合条目成员拥有相同本地成员名（称）的引用复合条目的每个成员进行标记。对引用复合条目的所有其他成员都不做标记。

由于一个搜索过滤器可能匹配若干个复合条目，因此最终的选择和标记将是单个匹配复合条目选择和标记的联合。

如果一个不是复合条目的匹配条目在其层次型选择中引用了一个复合条目，那么该复合条目的所有成员都被标记为参与成员。

有关该条目标记如何影响条目信息的返回在 7.6.4 中详述。

家族成员可以包装进一个family-information 派生的属性。如果在结果中只返回了一个单个复合条目成员，那么将不执行包装。不过，如果从读或修改条目操作返回了若干成员，那么将对这些成员进行包装。当一个搜索操作返回若干复合属性成员时，将对它们进行包装，除非设置了separateFamily-Members 搜索控制选项，在这种情况下，成员将作为单独条目返回。

当执行涉及复合条目的搜索操作时，对搜索操作有四个相关的阶段：

a) 每个关注条目中的家族成员组，如familyGrouping 所定义，在逻辑上在每个候选条目中考虑（即通过子集选择）。通过将所有的组属性汇集在一起，认为给定属性类型的所有属性值属于这个单个属性类型，即使它们来自不同的家族成员；

b) 过滤器适用于每个家族组合；如果过滤器满足组要求，那么复合条目满足过滤器要求，并考虑通过过滤器进行选择。对家族成员进行标记，如上所述；

c) 增加标记条目，通过EntryInformationSelection 中的familyReturn 来规定，以标记将返回的所有条目；

d) 如果在管理搜索规则中出现additionalControl 组件（见 GB/T 16264.2—2008 中 16.10.8），那么改变标记以及因此而返回的内容，作为处理所引用控制属性的结果。

8 绑定和解绑定操作

目录绑定和目录解绑定操作分别在 8.1 和 8.2 中定义，由 DUA 在访问目录的某个特定周期的开始和结束时使用。

8.1 目录绑定

8.1.1 目录绑定语法

目录绑定操作在访问目录的周期的开始使用。操作变元可以由请求方进行标记、加密或者标记和

加密(见 GB/T 16264.2—2008 中 17.3)。如果这样请求,那么目录可以对结果进行标记、加密或者标记和加密。

```
directoryBind OPERATION ::={
    ARGUMENT                DirectoryBindArgument
    RESULT                  DirectoryBindResult
    ERRORS                  { directoryBindError } }
DirectoryBindArgument ::=SET {
        Credentials         [0]  Credentials OPTIONAL,
        Versions            [1]  Versions DEFAULT {v1} }
Credentials ::=CHOICE {
        simple              [0]  SimpleCredentials,
        strong              [1]  StrongCredentials,
        externalProcedure   [2]  EXTERNAL,
        spkm                [3]  SpkmCredentials,
        sasl                [4]  SaslCredentials }
SimpleCredentials ::=SEQUENCE {
        Name          [0]  DistinguishedName,
        Validity      [1]  SET {
            time1       [0]     CHOICE {
                        utc     UTCTime,
                        gt      GeneralizedTime } OPTIONAL,
            time2       [1]     CHOICE {
                        utc     UTCTime,
                        gt      GeneralizedTime } OPTIONAL,
            random1     [2]     BIT STRING OPTIONAL,
            random2     [3]     BIT STRING OPTIONAL } OPTIONAL,
        password [2]    CHOICE {
            unprotected         OCTET STRING,
            protected           SIGNATURE {OCTET STRING} } OPTIONAL}
StrongCredentials ::=SET {
        certification-path          [0]  CertificationPath OPTIONAL,
        bind-token                  [1]  Token,
        name                        [2]  DistinguishedName OPTIONAL,
        attributeCertificationPath  [3]  AttributeCertificationPath OPTIONAL }
SpkmCredentials ::=CHOICE {
        req     [0]     SPKM-REQ,
        rep     [1]     SPKM-REP-TI }
SaslCredentials ::=SEQUENCE {
        Mechanism   [0]     DirectoryString { ub-saslMechanism },
        credentials [1]     OCTET STRING OPTIONAL,
        saslAbort   [2]     BOOLEAN DEFAULT FALSE }
Token ::=SIGNED { SEQUENCE {
        algorithm           [0]  AlgorithmIdentifier,
```

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

8.1.2 目录绑定变元

DirectoryBindArgument 的credentials 变元允许目录建立用户的身份。证书可以是simple 或strong 或外部定义的(externalProcedure)(如 ISO/IEC 9594-8:2005 中所述)。

如果使用simple,那么它包括一个name(总为客体的可辨别名)、一个可选的validity 以及一个可选的password。这提供了有限程度的安全。password 可以是unprotected,或者可以是protected(保护 1 或保护 2),如 ISO/IEC 9594-8:2005 中所述。validity 提供了time1、time2、random1 和random2 变元,其含义来自双边协议,可用于检测重放。在某些情况下,受保护的口令可以通过一个客体来检查,该客体只有在本地重建对其自身口令拷贝的保护并对结果与绑定变元(password)中的值进行比较后,才能知晓口令。在其他情况下,可能直接进行比较。

如果商定的版本为v2 或更高版本,那么将对time1 和time2 使用GeneralizedTime。当商定结果为v1 时,GeneralizedTime 的使用可以有碍与实现的交互,实现无法感知是可能选择UTCTime 还是可能选择GeneralizedTime。这是那些用于规定域的版本的责任,在这些域中将使用目录规范,例如描述概貌组,什么时候可以使用GeneralizedTime。UTCTime 将不用于描述超过 2049 年的日期。

如果使用strong,那么它包括一个bind-token、一个可选的certification-path(鉴别和鉴别权威部门交叉鉴别序列,如 ISO/IEC 9594-8:2005 中所述)以及请求方的name。这使得目录能够对建立联系的请求方身份进行验证,反之亦然。如果在绑定操作中使用StrongCredentials 或pkmCredentials,则传送有关身份和验证的信息。这使得能够对任何一个实体的身份进行验证,还使得能够使用已经建立的密码以及完整性密码密钥资料。

BindIntAlgorithm 和bindConfAlgorithm 组件用于商定密码算法,以便用于保护绑定中的后续操作。请求方包括一个按优先次序排列的支持算法清单。目录从清单中选择一个算法,它符合其自身的安全策略要求,并在响应中指示这一点。

完整性和机密性算法使用的会话密钥通过使用bindIntKeyInfo 和bindConfKeyInfo 字段来建立。通过产生一个适当长度的会话密钥,并用其他公钥进行加密,请求方和目录对会话密钥的选择都有影响。会话密钥是这两个组件的异或。注意,请求方可以将会话密钥的生成交给目录,在这种情况下,将从绑定变元中删去上述各字段。

注1：鉴别证书可以通过安全交换服务元素进行传送(见GB/T 16264.5—2008)，在这种情况下，它们将不出现在绑定变元或结果中。

如果对操作进行标记和加密，那么包含属性的属性证书(见ISO/IEC 9594-8:2005第12章)可以用于传送属性访问所需的清晰说明。attributeCertificationPath用于传送基于规则的访问控制的安全清晰说明，或者在属性证书中传送的其他属性，可选地，还有验证属性证书所需的证书。

绑定令牌的变元如下使用。algorithm是用于标记该信息的算法标识符。name是计划中接收方的名(称)。time参数包含令牌的终止时间。random数是一个应不同于各个未终止令牌的数，接收方可用之来检测重放攻击。

注2：当名(称)用在简单或增强证书中时，如果存在，可能使用可选的可辨别名。不过，如果不使用主可辨别名，那么基于名(称)的鉴别和访问控制可能无法如期望的那样开展工作。在成功处理经鉴别的BIND操作后，在BIND变元中无论使用什么名(称)，各绑定实体相互间都将知道其主可辨别名，以便在BIND起作用的情况下推动访问控制操作。

如果使用externalProcedure，那么正在使用的鉴别方案的语义将在目录规范范围之外。

当使用RFC 2222中规定的简单鉴别和安全层(SASL)时，使用sasl。如果通过值设为空字符串的SaslCredentials机制来调用directoryBind操作，那么将返回一个inappropriateAuthentication的SecurityError。

DirectoryBindArgument的versions变元用于确定DUA准备参与的服务的版本。值v1表示协议版本1，值v2表示协议版本2。如果在后续ModifyEntry操作中将传送alterValues或resetValue修改类型，或者需要一个非NULL的结果(见11.3)，那么将使用值v2。如果对增加条目、移除条目、修改条目、修改DN使用差错或结果标记，那么值将设为v2。

通过以下措施推动目录向未来版本的迁移：

a) 将接受和忽略DirectoryBindArgument的任何元素，而非本目录规范中定义的那些元素；

b) 将接受和忽略有关未定义的DirectoryBindArgument命名位(如版本)的各额外选项。

如果要求抢占响应鉴别，那么response组件用于承载一个随机数。

BindIntAlgorithm、bindKeyInfo、bindConfAlgorithm和bindConfKey组件用于承载保护绑定中后续操作所需的信息。

8.1.3 目录绑定结果

如果绑定请求成功，那么将返回一个结果。

DirectoryBindResult的credentials变元允许用户建立目录的身份。它允许将用于确定DSA(直接提供目录服务)的信息传送给DUA。其形式(即CHOICE)将与用户提供的形式相同。

DirectoryBindResult的versions参数用于指示DSA将实际提供哪个版本的服务(DUA请求的)。

8.1.4 目录绑定差错

如果绑定请求失败，那么将返回一个绑定差错。

directoryBindError的versions参数指示DSA支持哪个版本。

securityError或serviceError将按如下方式提供：

——securityError inappropriateAuthentication
　　invalidCredentials
　　blockedCredentials
——serviceError unavailable
　　saslBindInProgress

8.2 目录解绑定

访问目录周期结束之时的解绑定针对的是GB/T 16264.5—2008中7.6.4所规定的OSI环境以及GB/T 16264.5—2008中9.3.2中所规定的TCP/IP环境。

注：在解绑定时，尚未访问的所有分页结果将变得不可访问，应去除。

9 目录读操作

有两个“类似读”操作:read 和compare,分别在 9.1 和 9.2 中定义。为方便起见,在 9.3 中定义的abandon 操作将与这些操作分在一组。

9.1 读

9.1.1 读语法

读操作用于从一个显式标识的条目中提取信息。它还可用于验证可辨别名。操作变元可以由请求方进行标记(见 GB/T 16264.2—2008 中 17.3)。如果这样请求,那么目录可以对结果进行标记。

```
read OPERATION ::= {
        ARGUMENT        ReadArgument
        RESULT          ReadResult
        ERRORS          { attributeError | nameError | serviceError | referral | abandoned |
                        securityError }
        CODE            id-opcode-read }
    ReadArgument ::= OPTIONALLY-PROTECTED {
        SET {
            object                  [0] Name,
            selection               [1] EntryInformationSelection DEFAULT { },
            modifyRightsRequest     [2] BOOLEAN DEFAULT FALSE,
            COMPONENTS OF   CommonArguments } }
ReadResult ::= OPTIONALLY-PROTECTED {
        SET {
            entry           [0] EntryInformation,
            modifyRights    [1] ModifyRights OPTIONAL,
            COMPONENTS OF   CommonResults } }
ModifyRights ::= SET OF SEQUENCE {
        item            CHOICE {
            entry       [0]   NULL,
            attribute   [1]   AttributeType,
            value       [2]   AttributeValueAssertion },
        permission      [3]   BIT STRING { add (0),remove (1),rename (2),move (3) } }
```

9.1.2 读变元

Object 变元用于请求信息的客体条目。如果Name 涉及一个或多个别名,那么解除引用它们(除非相关的服务控制禁止之)。Name 可以是一个可替换的名(称),并可包括上下文信息,如 GB/T 16264.2—2008 中 9.3 中所述。

selection 变元指示自条目请求什么信息(见 7.6)。不过,不应假设返回的属性等同于请求的那些属性或限于请求的那些属性。

CommonArguments (见 7.3)包括有关适用于请求的服务控制和安全参数的规定。出于本操作的目的,sizeLimit 组件不相关,如果提供,将被忽略。如果请求方对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters (见 7.10)组件将包括在变元中。

modifyRightsRequest 变元用于请求将请求方的修改权限返回给条目及其属性。

9.1.3 读结果

如果请求成功,那么将返回结果。

条目结果参数持有请求的信息(见 7.7)。如果因EntryInformationSelection 中familyReturn 元素出现而提出要求,那么这可以包括家族信息。

如果通过modifyRightsRequest 变元提出请求,那么modifyRights 参数出现,用户对某些或所有的请求条目信息拥有修改特权,本地安全策略允许返回该信息。如果返回,那么为条目和selection 变元中规定的属性返回请求方的修改权限。参数包含以下内容:

——为entry、对每个请求的用户attribute(用户拥有增加或移除权限)、为每个返回的属性value(用户增加或移除它的权限不同于对应属性的那些权限)返回一个SET 的元素。

——返回的permission 指示用户在条目上实施的哪些操作或行为将取得成功。在一个条目的情况下,remove 指示RemoveEntry 操作将取得成功;rename 指示,如果newSuperior 参数不存在,那么ModifyDN 将取得成功;move 指示,如果newSuperior 参数出现,那么ModifyDN 和未改变的 RDN 将取得成功。在属性和值的情况下,add 指示增加属性或值的ModifyEntry 将取得成功;remove 指示,移除属性或值的ModifyEntry 将取得成功。

注:将条目移至一个新上级的操作还可能依赖于与新上级关联的许可(例如,通过basic-accesscontrol)。当确定permission 时,这些将被忽略。

CommonResults(见 7.4)包括适用于响应的安全参数。如果目录对结果进行标记、加密或者标记和加密,那么SecurityParameters 组件(见 7.10)将包括在结果中。

9.1.4 读差错

如果请求失败,那么将报告其中的一个列出差错。如果无法返回任何一个明确列于selection 中的属性,那么将报告一个带问题noSuchAttributeOrValue 的attributeError。将报告其他差错的情况在第12 章中定义。

9.1.5 基本访问控制的读操作决策点

如果rule-based-access-control 也应用,那么有关basic-access-control 应用的次序问题将是一个本地问题,除非有任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在读的条目起作用,那么应用以下访问控制序列:

a) 对正在读的条目需要读许可。如果未赋予许可,那么依据 7.11.1.3,操作失败。

b) 如果selection 的infoTypes 元素规定只能返回属性类型,那么对每个将要返回的属性类型,需要读许可。如果未赋予许可,那么从ReadResult 中删去属性类型。如果作为应用这些控制的结果,没有返回任何属性信息,那么依据 9.1.5.1,整个操作失败。

c) 如果selection 的infoTypes 元素规定返回属性类型和值,那么对每个将要返回的属性类型和值,需要读许可。如果对属性类型未赋予许可,那么从ReadResult 中删去属性。如果对属性值未赋予许可,那么从其对应的属性中删去值。在未将许可赋予属性内任何值的情况下,返回一个包含空SET OF AttributeValue 的Attribute 元素。如果作为应用这些控制的结果,没有返回任何属性信息,那么依据 9.1.5.1,整个操作失败。

注:允许 DAP 读操作的特权在 LDAP 环境中可能不起作用,当中需要得到浏览许可,以便支持相当的读服务。

9.1.5.1 差错返回

如果操作失败,如 9.1.5 中 b)和 c)定义,那么有效的差错返回为以下之一:

a) 如果规定了一个无限制的选项(即allUserAttributes 或allOperationalAttributes),那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError。

b) 否则,如果规定了一个select 选项(在attributes 中与/或在extraAttributes 中),那么如果为任

何选定的属性赋予了*DiscloseOnError* 许可，那么将返回一个带问题insufficientAccessRights或noInformation 的securityError。否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

9.1.5.2 不泄露不完整的结果

如果在EntryInformation 中返回一个不完整的结果，即因适用的访问控制而删去了某些属性或属性值，如果*DiscloseOnError* 许可赋给至少一个结果中保留的属性类型，或者赋给至少一个结果中保留的属性值(对该属性类型赋予了*读*许可)，那么incompleteEntry 元素将被设为 TRUE。

9.1.6 基于规则的访问控制的读操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在读的条目起作用，那么应用以下访问控制：

a) 如果在rule-based-access-control 下拒绝条目级访问，那么依据 7.11.2.4，操作失败，返回一个带问题noSuchObject 的nameError。

b) 如果在basic-access-control 方案下不允许访问条目，如 9.1.5a)所述，那么依据 7.11.1.3，操作失败。

c) 如果selection 的infoTypes 元素规定只返回属性类型，那么如果在rule-based-accesscontrol 下，未准予访问该类型的所有属性值，那么从ReadResult 中删去属性类型。如果作为应用这些控制的结果，未返回任何属性信息，那么依据 9.1.5.1 b)，整个操作失败，返回一个带问题noSuchAttributeOrValue 的attributeError。

d) 如果selection 的infoTypes 元素规定只返回属性类型，那么应用basic-access-control，如 9.1.5b)所述。

e) 在rule-based-access-control 下，如果selection 的infoTypes 元素规定返回属性类型和值，那么对每个将要返回的属性值，将准予访问。如果未准予访问某个属性值，那么从其对应的属性中删去该属性值。在未准予访问某个属性中任何属性值的情况下，从ReadResult 中删去整个属性。如果作为应用这些控制的结果，未返回任何属性信息，那么整个操作失败，返回一个带问题noSuchAttributeOrValue 的attributeError。

f) 应用basic-access-control，如 9.1.5c)所述。

g) 按 7.11.2.2 定义确定在操作结果中返回的名(称)。

9.2 比较

9.2.1 比较语法

比较操作用于一个值(作为请求变元提供)与某个特定客体条目中某个特定属性类型值的比较。操作变元可以由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 中 17.3)。如果这样请求，那么目录可以对结果进行标记、加密或者标记和加密。

除了multiStrand，可以使用任何familyGrouping 值，所有成组家族成员的属性都将在与假设的属性值断言的比较中使用。如果familyGrouping 规定了multiStrand，那么采用compoundEntry。

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

```
    CODE                id-opcode-compare }
CompareArgument ::=OPTIONALLY-PROTECTED {
        SET {
            object          [0]     Name,
            purported       [1]     AttributeValueAssertion,
            COMPONENTS OF           CommonArguments } }
CompareResult ::=OPTIONALLY-PROTECTED {
        SET {
            name                    Name OPTIONAL,
            matched         [0]     BOOLEAN,
            fromEntry       [1]     BOOLEAN DEFAULT TRUE,
            matchedSubtype  [2]     AttributeType OPTIONAL,
            COMPONENTS OF           CommonResults } }
```

9.2.2 比较变元

object 变元为所考虑的特殊客体条目的名(称)。如果Name 涉及一个或多个别名,那么解除引用(除非相关的服务控制禁止之)。Name 可以是一个可选的名(称),并可包括上下文信息,如 GB/T 16264.2—2008 中 9.3 中所述。

purported 变元用于确定将要与条目中属性类型和值进行比较的属性类型和值。如果条目持有假设的属性类型或其子类型之一,或者存在一个为假设的属性类型或其子类型之一的共同条目属性(见 7.6),并且如果存在一个匹配假设值的属性值(利用属性的 equality 匹配规则),那么比较结果为 TRUE。

注:compare 请求无法满足变元中所规定的属性类型的友人属性类型的要求。

如果属性值断言中包括上下文断言,那么将只对那些满足所有给定上下文断言要求的值尝试匹配,如 GB/T 16264.2—2008 中 8.9.2 所述。如果在属性值断言中未包括任何上下文断言,那么将应用缺省上下文断言,如 GB/T 16264.2—2008 中 8.9.2.2 所述。

CommonArguments(见 7.3)包括有关服务控制和适用于请求的安全参数的规定。出于该操作的目的,sizeLimit 组件不相关,如果提供,将被忽略。如果请求方对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters(见 7.10)组件将包括在变元中。

9.2.3 比较结果

如果请求成功(即真实地完成了比较),那么将返回结果。

name 为条目的可辨别名或条目的别名,如 7.7 所述。只有当别名解除引用时、当 RDN 已解析为主要 RDN 时,或者当内容选择已应用时、当将要返回的名(称)不同于操作变元中所提供的Object 名(称)时,它才出现。

matched 结果参数持有比较结果。如果对值进行了比较并匹配,那么参数取 TRUE 值,如果不是这样,那么取 FALSE 值。

如果fromEntry 为TRUE,那么信息与条目进行比较;如果为 FALSE,那么信息与拷贝进行比较。

只有当匹配结果为 TRUE 并且因假设属性的子类型匹配而使匹配取得成功时,matchedSubtype 参数才出现。如果多个这样的子类型可用,那么返回层次中最高的那个。

CommonResults(见 7.4)包括适用于响应的安全参数。如果目录对结果进行签名、加密或者签名和加密,那么SecurityParameters 组件(见 7.10)将包括在结果中。

9.2.4 比较差错

如果请求失败,那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。

9.2.5 基本访问控制的比较操作决策点

如果rule-based-access-control 也应用，那么有关basic-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在比较的条目起作用，那么应用以下访问控制序列：

a) 对将要比较的条目需要读许可。如果不赋予该许可，那么依据 7.11.1.3，该操作失败；

b) 对正在比较的属性需要比较许可。如果不赋予该许可，那么依据 9.2.5.1，该操作失败；

c) 如果在正在比较的属性中存在一个匹配purported 变元的值，并且赋予了比较许可，那么操作在CompareResult 的matched 结果参数中将返回值TURE。否则，操作将返回值FALSE。

9.2.5.1 差错返回

如果操作失败，如 9.2.5 中 b)定义，那么有效的差错返回是如下之一：如果将*DiscloseOnError* 许可赋予了正在比较的属性，那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError；否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

9.2.6 基于规则的访问控制的比较操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在比较的条目起作用，那么应用以下访问控制：

a) 如果在rule-based-access-control 下拒绝条目级访问，那么依据 7.11.2.4，操作失败，返回一个带问题noSuchObject 的nameError；

b) 如果在based-access-control 方案下不允许访问条目，如 9.2.5 中 a)所述，那么依据 7.11.1.3，操作失败；

c) 如果访问未赋予正在比较的属性值，那么目录将按以下方式开展工作，即仿佛属性值没有出现；

d) 应用basic-access-control，如 9.2.5 中 b)和 c)所述；

e) 按 7.11.2.2 定义确定在操作结果中返回的名(称)。

9.3 放弃

如果用户对结果不再关注，那么可以利用abandon 操作放弃查询目录的操作。操作变元可以由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 的 17.3)。如果这样请求，那么目录可以对结果进行标记、加密或者标记和加密。

```
abandon OPERATION ::={
    ARGUMENT        AbandonArgument
    RESULT          AbandonResult
    ERRORS          { abandonFailed }
    CODE            id-opcode-abandon }
AbandonArgument ::=OPTIONALLY-PROTECTED-SEQ {
    SEQUENCE {
        invokeID            [0]    InvokeId } }
AbandonResult ::=CHOICE {
    null                NULL,
```

```
information       OPTIONALLY-PROTECTED-SEQ {
                      SEQUENCE {
                              invokeID                InvokeId,
                              COMPONENTS OF           CommonResultsSeq } } }
```

有单个变元invokeID,用于确定将要放弃的操作。InvokeID 的值等于用于调用将要放弃的操作的invokeID。

如果请求成功,那么将返回一个结果。如果目录对该结果进行标记、加密或者标记和加密,那么CommonResultsSeq(见 7.4)的SecurityParameters 组件(见 7.10)将包括在结果中。如果目录不对该操作的结果进行标记,那么不随结果传送任何信息。最初的操作将因abandoned 差错而失败。

如果请求失败,那么将报告abandonFailed 差错。作为一个本地问题,DSA 可以选择不放弃操作,而后返回abandonFailed 差错。该差错在 12.3 中进行描述。

放弃仅适用于查询操作,即读、比较、列表和搜索操作。

DSA 可以在本地放弃一个操作。如果 DSA 已将操作链接或多点传送至其他 DSA,那么它可以依次对它们进行查询,以便放弃操作。

10 目录搜索操作

有两个"类似搜索"操作:列表和搜索,分别在 10.1 和 10.2 中定义。

10.1 列表

10.1.1 列表语法

列表操作用于获取一个明确确定的条目的直接下级。在某些情况下,返回的列表是不完整的。操作变元可以由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 的 17.3)。如果这样请求,那么目录可以对结果进行标记、加密或者标记和加密。

```
list    OPERATION ::={
        ARGUMENT        ListArgument
        RESULT          ListResult
        ERRORS          { nameError|serviceError|referral|abandoned|securityError }
        CODE            id-opcode-list }
ListArgument ::=OPTIONALLY-PROTECTED {
        SET {
                object                  [0]  Name,
                pagedResults            [1]  PagedResultsRequest OPTIONAL,
                listFamily              [2]  BOOLEAN DEFAULT FALSE,
                COMPONENTS OF                CommonArguments } }
ListResult ::=OPTIONALLY-PROTECTED {
        CHOICE {
                listInfo                     SET {
                        name                              Name OPTIONAL,
                        subordinates                [1]   SET OF SEQUENCE {
                                rdn                           RelativeDistinguishedName,
                                aliasEntry                [0] BOOLEAN DEFAULT FALSE,
                                fromEntry                 [1] BOOLEAN DEFAULT TRUE },
                        partialOutcomeQualifier     [2]   PartialOutcomeQualifier OPTIONAL,
                        COMPONENTS OF                     CommonResults },
```

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

10.1.2 列表变元

object 变元用于确定客体条目(或可能是根),将列出其直接下级。如果Name 涉及一个或多个别名,那么解除引用它们(除非相关的服务控制禁止这么做)。Name 可以是一个可选的名(称),并可以包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

pagedResults 变元用于请求逐页返回的操作结果,如 7.9 所述。

如果listFamily 为TRUE,并且object 为祖(条目),那么列出的下级取自直接的下级家族成员;不包括任何其他下级。否则,列出的下级只能取自不是家族成员的直接下级。

CommonArguments (见 7.3)包括适用于请求的服务控制规定。如果请求方将对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters 组件(见 7.10)将包括在变元中。

10.1.3 列表结果

依据访问控制,如果找到了object,而不管是否返回下级信息,那么请求成功。

name 为条目的可辨别名或条目的别名,如 7.7 所述。只有当别名解除引用时、当 RDN 已解析为主要 RDN 时,或者当内容选择已应用时、当将要返回的名(称)不同于操作变元中所提供的object 名(称)时,它才出现。

如果有的话,subordinates 参数用于传送命名条目直接下级上的信息。如果任何下级条目为别名,那么它们将不被解除引用。

rdn 参数为下级的相对可辨别名。这可能受 7.7 中为Name 所述的上下文影响。

fromEntry 参数用于指示信息是取自条目(TRUE)还是条目的一个拷贝(FALSE)。

aliasEntry 参数用于指示下级条目是一个别名条目(TRUE)还是不是一个别名条目(FALSE)。

partialOutcomeQualifier 由九个如下所述的子组件组成。无论何时,当由于时间限制、大小限制或管理限制等问题、由于未开发 DIT 区域、由于某些重要的扩展不可用、由于收到了一个位置的差错、由于返回分页结果、由于指出一个过度规定的过滤器、由于返回一个或多个通告属性、或者由于操作结果是一个流结果并且该响应不是结果的最后一个响应,而使结果不完整时,将出现该参数。

a) LimitProblem 参数用于指出是否已经超出了时间限制、大小限制或管理限制。返回的结果为当达到限度时可用的那些结果。

b) 如果没有探索到 DIT 区域,那么unexplored 参数将出现。其信息允许 DUA 通过联系其他访问点(如果它这么选择的话)来继续处理列表操作。参数包括一系列(可能为空)Continuation-References,每个包括基本客体(在其上可以进行操作)的名(称)、OperationProgress 的适当

值、一系列访问点(在其上可以进一步进行请求)。返回的ContinuationReferences 将处于在操作服务控制请求的转向推荐范围内。见 12.6。

c) unavailableCriticalExtensions 参数指出,如果出现,在目录的某些部分,一个或多个临界扩展是不可用的。

d) unknownErrors 参数用于返回未知的差错类型或在操作处理中自其他 DSA 接收的参数。SET 的每个成员都包含一个这样的未知差错。见 GB/T 16264.5—2008 中 12.2.4。

e) 当 DUA 已请求分页结果并且 DSA 未返回所有的可用结果时,queryReference 参数将出现。见 7.9。当 DSA 能够确定对用户有效的所有结果都已返回时,它将不存在(即,它不是一个应用访问控制的结果)。

f) overspecFilter 组件只与搜索操作一起使用,当作为过度规定过滤的结果,返回的搜索结果为空时,虽然存在候选的条目,它们只匹配于部分过滤器,或者只大致匹配于过滤器。只有当搜索请求包括checkOverspecified 项并且目录能够确定过滤器过度规定了时,才返回它。它包括在search 变元中提供的过滤器,利用成功匹配的过滤器的那些元素,删去某些目。产生overspecFilter 的实际规程是一个本地问题。

注 1:分布式目录中适当overspecFilter 的返回有待进一步研究。

g) notification 参数可以用来发送差错结果限定,并可针对使用的搜索操作,返回一个proposedRelaxation 属性(见 GB/T 16264.6—2008 中 5.12.15,它提供了一种张弛策略,可供用户使用。在这种情况下,可以提供MRMapping 元素序列,它将用于影响张弛(或紧缩)策略,由有关的搜索规则规定。

注 2:notification 中sequence-of Attribute 的次序并不重要。

h) entryCount 参数只与search 结果有关,并且如果出现,那么它将对满足搜索准则要求的条目数量给出一个最佳的估计。该子组件将出现,当且仅当:

- 在搜索变元中或者通过管理搜索规则设置了entryCount 搜索控制选项;
- 如果已经请求了分页结果或者超出了大小限制;以及
- 如果至少一个参数 DSA 支持该特性。

当entryCount 子组件出现时,如果所有执行 DSA 都支持该特性,并且如果所有符合要求的 DSA 都参与了操作,那么将采用bestEstimate 或准确选择。如果所有参与的 DSA 都能提供一个准确的计数,那么将采用准确的选择,否则将采用bestEstimate 选择。如果不是所有符合要求的 DSA 都参与了操作,或者部分参与的 DSA 不支持entryCount 参数,那么将采用lowEstimate 选择。复合条目的家族成员只当作是一个单个条目。

i) streamedResult 参数指示,当出现并为TRUE 时,DSA 发送一个流结果并且该响应不是结果的最后响应。如果不存在或以 FALSE 出现,该参数指示该响应是流结果的最后响应或者它是一个非流响应。流结果中的每个响应将用相同的invokeId 进行确定。

如果遇到限制问题,它将导致在PartialOutcomeQualifier 中使用limitProblem 元素,那么该组件将在所有作为分页结果集的一部分而提供的所有后续结果中予以重复。

注 3:流结果中的每个响应都将利用同一invokeID 进行确定。这样,只有 IDM 目录协议可以使用该选项,如 GB/T 16264.5—2008 所规定。

当 DUA 已经提出了标记保护请求时,或者如果出于其他原因致使目录无法关联信息,那么uncorrelatedListInfo 参数可以包含若干源自目录不同组件并由之标记的结果参数集。如果在链接中没有任何 DSA 能够关联所有的结果,那么 DUA 将从各种不同的片断中组装实际的结果。

CommonResults(见 7.4)包括适用于响应的安全参数。如果目录对结果进行标记、加密或者标记和加密,那么SecurityParameters 组件(见 7.10)将包括在结果中。

10.1.4 列表差错

如果请求失败,那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行

定义。

10.1.5 基本访问控制的列表操作决策点

如果rule-based-access-control 也应用，那么有关basic-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在执行list 操作的那部分 DIB 起作用，那么应用以下访问控制序列：

a) 对由object 变元确定的条目不需要任何特殊的许可；

b) 对每个将在subordinates 中返回一个RelativeDistinguishedName 的直接下级，对该条目需要浏览和*ReturnDN* 许可。将忽略那些未赋予许可的条目。如果作为应用这些控制的结果，没有返回任何下级信息（PartialOutcomeQualifier 中的任何ContinuationReferences 除外），并且如果未将*DiscloseOnError* 许可赋予给由object 变元确定的条目，那么操作失败，并将返回一个带问题noSuchObject 的nameError。matched 元素或者包含下一个上级条目的名（称），该条目赋予了*DiscloseOnError* 许可，或者包含 DIT 根的名（称）（即一个空RDNSequence）。否则，操作成功，但它不传送任何下级信息（PartialOutcomeQualifie 中的任何ContinuationReferences 除外）。

注 1：在返回nameError 的情况下，未访问所有上级条目的 DSA 可能使用空RDNSequence。

注 2：安全策略可以防止泄漏下级信息，否则将作为PartialOutcomeQualifier 中的ContinuationReferences 予以传送。如果这样一个策略发挥作用，并且如果 DUA 通过规定chainingProhibited 来约束服务，那么目录可能返回一个带问题chainingRequired 的serviceError。否则将接着执行上面 b）中所述的规程。

注 3：安全策略可以防止目录指示一个列出的下级条目是一个别名条目。例如，如果 DUA 未将读访问赋予别名条目、其包含的objectClass 属性和值alias，那么目录可能从ListResult 中删去subordinates 的aliasEntry 组件，或者将之设为FALSE。

注 4：如果未将*DiscloseOnError* 许可赋予object 变元确定的条目，那么不应返回指示limitProblem 或unavailableCriticalExtensions 的partialOutcomeQualifier，原因是它可能对本条目的安全造成危害。

10.1.6 基于规则的访问控制的列表操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在执行List 操作的那部分 DIB 起作用，那么应用以下访问控制序列：

a) 如果对由object 变元确定的条目拒绝基于规则的条目级许可，那么将依据 7.11.2.4 返回带问题noSuchObject 的nameError。

b) 对每个将在subordinates 中返回RelativeDistinguishedName 的直接下级，务必将基于规则的RDN 许可赋予给该条目。忽略未准予访问的各条目。

c) 应用basic-access-control，如 10.1.5 所述。

10.2 搜索

10.2.1 搜索语法

搜索操作对关注的条目搜索目录的一个或多个部分，并从这些条目返回选定的信息。操作变元可以由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 的中 17.3）。如果这样请求，那么目录可以对结果进行标记、加密或者标记和加密。

```
search OPERATION ::= {
        ARGUMENT        SearchArgument
```

```
        RESULT          SearchResult
        ERRORS          { attributeError|nameError|serviceError|referral|abandoned|
                        securityError }
        CODE            id-opcode-search }
SearchArgument ::=OPTIONALLY-PROTECTED {
        SET {
            baseObject          [0]   Name,
            subset              [1]   INTEGER {
                baseObject(0),oneLevel (1),wholeSubtree(2)} DEFAULT baseObject,
            filter              [2]   Filter DEFAULT and: { },
            searchAliases       [3]   BOOLEAN DEFAULT TRUE,
            selection           [4]   EntryInformationSelection DEFAULT { },
            pagedResults        [5]   PagedResultsRequest OPTIONAL,
            matchedValuesOnly   [6]   BOOLEAN DEFAULT FALSE,
            extendedFilter      [7]   Filter OPTIONAL,
            checkOverspecified  [8]   BOOLEAN DEFAULT FALSE,
            relaxation          [9]   RelaxationPolicy      OPTIONAL,
            extendedArea        [10]  INTEGER               OPTIONAL,
            hierarchySelections [11]  HierarchySelections DEFAULT { self },
            searchControlOptions [12] SearchControlOptions  DEFAULT { searchAliases },
            joinArguments       [13]  SEQUENCE SIZE (1..MAX) OF JoinArgument OPTIONAL,
            joinType            [14]  ENUMERATED {
                innerJoin(0),leftOuterJoin(1),fullOuterJoin (2) } DEFAULTleftOuterJoin,
            COMPONENTS OF     CommonArguments } }
HierarchySelections ::=BIT STRING {
        self            (0),
        children        (1),
        parent          (2),
        hierarchy       (3),
        top             (4),
        subtree         (5),
        siblings        (6),
        siblingChildren (7),
        siblingSubtree  (8),
        all             (9)}
SearchControlOptions ::=BIT STRING {
        searchAliases           (0),
        matchedValuesOnly       (1),
        checkOverspecified      (2),
        performExactly          (3),
        includeAllAreas         (4),
        noSystemRelaxation      (5).
        dnAttribute             (6),
```

```
        matchOnResidualName          (7),
        entryCount                   (8),
        useSubset                    (9),
        separateFamilyMembers        (10),
        searchFamily                 (11) }
JoinArgument ::=SEQUENCE {
        joinBaseObject          [0]  Name,
        domainLocalID           [1]  DomainLocalID OPTIONAL,
        joinSubset              [2]  ENUMERATED {
              baseObject(0),oneLevel(1),wholeSubtree(2)} DEFAULT baseObject,
        joinFilter              [3]  Filter OPTIONAL,
        joinAttributes          [4]  SEQUENCE SIZE (1..MAX) OF JoinAttPair OPTIONAL,
        joinSelection           [5]  EntryInformationSelection }
DomainLocalID ::=DirectoryString { ub-domainLocalID }
```

DomainLocalID 是一个字符束，在本地唯一确定一个部分持有另一个 DIT 的远程域。

注：该字符束在本地定义，无需由任何注册机构进行注册。

```
JoinAttPair ::=SEQUENCE {
        baseAtt          AttributeType,
        joinAtt          AttributeType,
        joinContext      SEQUENCE SIZE (1..MAX) OF JoinContextType OPTIONAL }
JoinContextType ::=CONTEXT.&id({SupportedContexts})
SearchResult ::=OPTIONALLY-PROTECTED {
            CHOICE {
                SearchInfo                    SET {
                    name                          Name OPTIONAL,
                    entries                  [0]  SET OF EntryInformation,
                    partialOutcomeQualifier  [2]  PartialOutcomeQualifier OP-
                    TIONAL,
                    altMatching              [3]  BOOLEAN DEFAULT
                    FALSE,
                    COMPONENTS OF                 CommonResults },
                uncorrelatedSearchInfo       [0]  SET OF SearchResult } }
```

10.2.2 搜索变元

baseObject 变元用于标识与将要发生的主搜索相关的客体条目(或可能的话为根条目)。baseObject 可以是一个可选的名(称)，并包括上下文信息，如 GB/T 16264.2—2008 中 9.3 所述。

subset 变元指示主要搜索是否应用于：

a) 只有baseObject；

b) 只有基本客体的直接下级(oneLevel)；

c) 基本客体及其所有下级(wholeSubtree)。

如果基本客体是一个普通条目，那么依据subset 规定，认为复合条目是单个条目。如果基本客体是复合条目的祖(条目)，那么searchFamily 搜索控制选项将控制准确的行为。如果基本客体是一个子家族成员，那么认为家族成员是单个条目。

filter 变元用于从主要搜索空间中去除不关注的条目。只在符合过滤器要求的条目上返回信息(见 7.8)。在出现基本的用户提供的或搜索规则提供的张弛策略情况下，将以要求的匹配规则替换，在第一时间对过滤器进行评估。

在出现用户提供的或搜索规则提供的张弛策略情况下，或者在二者都出现的情况下，如果返回的结果比最低的要求还要少，那么将对过滤器重新进行评估，利用适当的张弛策略（见 7.8 和以下内容，它们有关 SearchArgument 的张弛元素），逐步递增，直至有足够多的条目或者没有更多的张弛策略可定义。同样，如果返回的结果比最高的要求还要多，那么也将对过滤器重新进行评估，利用适当的张弛策略，逐步递增，直至有足够少的条目或者没有更多的紧缩策略可定义。

注 1：如果未提供搜索规则的张弛策略，那么用户可能需要对过滤器进行简化，并做再次测试，或者可选地定义一个用户定义的张弛策略。

CommonArguments 的familyGrouping 组件用于在应用过滤器之前，在逻辑上将各条目合并进一个家族中，如 7.3.2 和 7.8.3 所述。

当找到基本客体后，依据dontDereferenceAliases 服务控制设置，别名将被解除引用。基本客体下的别名将依据searchAliases 参数设置，在搜索期间被解除引用。如果searchAliases 参数为TRUE，那么别名将被解除引用，如果参数为FALSE，那么别名将不被解除引用。如果searchAliases 参数为TRUE，那么将在别名条目的子树中继续进行搜索。

selection 变元指示自条目请求什么信息（见 7.6）。然而，不应假设返回的属性等同于请求的那些属性或限于请求的那些属性。

注 2：出于分布式操作的目的，用于协调相关条目分布式操作的 DSA（即已完成对包含joinArguments 的搜索变元的名（称）解析，并需要从非内部来源中获得一个潜在相关的条目集）需要超越 DAP 提供的、带attributeTypesAndValues的infoTypes，并且在选择需要利用分布式操作返回的属性中，还需要包括联合属性（即由JoinArgument . joinAttributes 中JoinAttPair . joinAtt 规定的集合中的属性）。不过，如果infoTypes 值为attributeTypesOnly，那么通过协调 DSA 返回给用户的条目和派生的条目将删去 DAP 返回信息中的属性值，并将因此依据最初的用户请求返回EntryInformation。

pagedResults 变元用于请求应逐页返回操作结果，如 7.9 所述。

matchedValuesOnly 变元用于指出将从返回的条目信息中删去某些属性值。尤其是，当返回的属性是多值的，并且某些但不是所有的属性值都对搜索过滤器起作用，以其最后的有效形式（即考虑张弛的匹配规则）通过present 之外的过滤器项返回 TRUE，那么从返回的目信息中删去不那么起作用的值。

如果在search 变元中规定了matchedValuesOnly 变元，那么对将要返回的属性应用以下逻辑处理过程：

a) 如果过滤器由一个过滤器项组成，那么应用以下规则：
   1) 如果过滤器项类型为present，那么matchedValuesOnly 变元对该过滤器项中的属性不起作用。
   2) 如果过滤器项类型为equality、substrings、greaterOrEqual、lessOrEqual、approximateMatch、contextPresent 或extensibleMatch，并且对属性的断言不为 TRUE，那么matchedValuesOnly变元对该属性不起作用。如果断言为 TRUE，那么将从返回的条目信息中删去不匹配过滤器项的该属性的值。
   3) 如果对过滤器项求反，那么matchedValuesOnly 变元对该属性不起作用。
b) 如果过滤器是复杂的（包括多个过滤器项），那么应用以下规则：
   1) 如果过滤器包含一个求反（即not）过滤器，那么matchedValuesOnly 变元对求反过滤器中的任何属性都不起作用。
      注 3：这还适用于嵌套的、求反的过滤器。
   2) matchedValuesOnly 变元对或（即 or）过滤器的任何元素的属性都不起作用，评估为 FALSE 或 UNDEFINED。
   3) 对一个在过滤器中出现多次的属性只需其中一次出现评估为 TRUE，如上面 a）的 2）所述，对有效的 matchedValuesOnly 变元，即一次有效将超越一次或多次忽略。
   4) 对 or 过滤器中的每个过滤器都应评估 matchedValuesOnly，即使过滤器的真值可以在彻

底评估完成之前确定。

在混合版本情况下，使用extendedFilter 变元来规定上述中的一个可选过滤器。当该变元出现时，filter 变元(如果有的话)将被第 2 版本和后续版本的系统所忽略。extendedFilter 总被第 1 版本系统所忽略。搜索张弛策略仅用于filter。

注 4：通过包括两个过滤器，在搜索请求的分布式处理中，DUA 就可以规定第 1 版本系统使用一个过滤器，第 2 版本和后续版本系统使用另一个不同的过滤器。第 1 版本系统不支持属性多态性或匹配规则断言。

如果搜索操作的结果为空并且目录能够确定这是因过滤器过度规定而造成的，那么将用checkOverspecified 变元来请求目录返回一个partialOutcomeQualifier 中的overspecFilter 项。

可以用relaxation 组件来规定一个用户提供的RelaxationPolicy，利用 GB/T 16264.2—2008 中 16.10 中定义的结构。

依据管理搜索规则，如果替换将引起搜索无效，那么search 请求规定的替换将不在服务特定管理区域内执行。当替换匹配规则出现以下情况时，将与管理搜索规则发生冲突：

a) 从search 过滤器中有效地移除一个或多个过滤器项；或者

b) 与属性类型的matchingUse 规定发生冲突(见 GB/T 16264.2—2008 中 16.10.2)。

注 5：nullMatch 匹配规则可以将一个或多个过滤器项从过滤器中移除。当使用该匹配规则时，管理搜索规则可能冲突。

如果在服务特定管理区域外执行搜索操作，或者如果管理搜索规则未提供RelaxationPolicy 组件，那么应用用户提供的RelaxationPolicy，如 GB/T 16264.2—2008 中 16.10.7 所述。当搜索规则提供的RelaxationPolicy 也出现时，依据以下规程，实施结合：

a) 搜索规则规定的基本替换策略，如果有的话，在搜索确认过程中应用。因此而进行管理搜索规则规定的、可能的基本替换。

b) 在search 请求中规定的基本替换和基于映射的映射，如果出现的话，将应用。不过，不应用将引起管理搜索规则冲突的基本替换，而是忽略之。在这种情况下，oldMatchingRule 值(如果提供了的话)适用于基本的匹配规则，即在搜索规则应用的基本替换策略不存在的情况下，它将适用。

c) 张弛/紧缩替换，如果有的话，在search 请求中规定，而后与任何规定的、基于映射的匹配一起使用，依据的是在 GB/T 16264.2—2008 中 16.10.7 中规定的规则。如果在任何引起与管理搜索规则不一致的点遇到替换匹配规则，那么彻底放弃该特殊替换，以及任何由search 请求为该属性类型规定的进一步的替换。如果在该过程中，search 请求中规定的minimum 或maximum 规定得到了满足，那么停止该过程。

d) 应用管理搜索规则提供的张弛或紧缩替换，例外是，对已执行了张弛或紧缩替换的属性类型，不进行任何替换。也就是说，进一步的张弛或紧缩替换只适用于到目前为止尚未进行张弛或紧缩替换的属性类型的匹配规则。在这部分过程中，将继续使用search 请求中的maximum 或minimum 规定，而不使用那些在管理搜索规则中规定的规定。

如果在search 请求中规定的替换提议了一个不支持的匹配规则，那么现有的匹配规则将继续发挥作用。如果该策略无法产生一个支持的匹配规则，那么过滤器项被评估为 UNDEFINED。

用户可以提议系统通过规定哑匹配规则systemProposedMatch 来提供某种张弛或紧缩。

extendedArea 组件用于指示张弛的程度(如果大于 0 的话)或者紧缩的程度(如果小于 0 的话)。如果该组件出现，那么它对张弛或紧缩有影响，如 GB/T 16264.2—2008 中 16.10.7 所述。

hierarchySelection 搜索控制通过一个比特束来规定将要在每个匹配条目层次型组中执行的层次型选择。对不是层次型组一部分的匹配条目将忽略之。如果匹配一个层次中的若干条目，那么层次型选择将不产生返回多次的相同条目。如果该搜索控制不出现，那么不执行任何层次型选择。当出现时，以下选择可能是单独的或结合的：

a) self 指示，应从匹配条目返回条目信息。如果这是唯一选择，那么它对应的是不进行任何层次型选择。

b) children 指示，对每个匹配的条目，如果有的话，从每个匹配条目的所有直接层次下级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回。

c) parent 指示，对每个匹配的条目，如果有的话，从每个匹配条目的所有直接层次上级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回。

d) hierarchy 指示，对每个匹配的条目，从所有层次上级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回。

e) top 指示，对每个匹配的条目，从层次顶层返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回，除非匹配条目是顶层条目。

f) subtree 指示，对每个匹配的条目，如果有的话，从其所有层次下级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回。

g) siblings 指示，对每个匹配的条目，从所有层次同胞返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目返回。

h) siblingChildren 指示，对每个匹配的条目，从所有层次同胞的直接层次下级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目及其同胞返回。

i) siblingSubtree 指示，对每个匹配的条目，从所有层次同胞的所有下级返回条目信息。如果这是唯一设置，那么没有任何信息从匹配条目及其同胞返回。

j) all 指示，对每个匹配的条目，从层次家族的所有条目返回条目信息。

searchControlOptions 组件只包含适用于搜索操作的控制选项。该组件拥有语义等同于搜索变元布尔类型组件语义的指示器。一个支持服务管理扩展的实现方案将支持该组件。一个发送支持的实现方案例如一个 DUA)在设置布尔类型组件之外还将设置该组件的各对应位(除非应用缺省值)。如果一个支持 DSA 的实现方案用该组件接收一个search 请求，那么它将忽略请求中的布尔类型组件。如果在请求中不存在该组件，那么缺省设置将被理解为重新设置所有位，除非如下所述：

a) searchAliases 搜索控制选项是对searchAliases 搜索变元组件的替换。如果设置了该位，那么它对应值为TRUE 的searchAliases 组件。如果searchControlOptions 组件不存在，那么缺省值取决于searchAliases 组件，即如果searchAliases 组件不存在或者设为 TRUE，那么该位缺省为设置值。

b) matchedValuesOnly 搜索控制选项是对matchedValuesOnly 搜索变元组件的替换。如果设置了该位，那么它对应值为TRUE 的matchedValuesOnly 组件。如果searchControlOptions 组件不存在，那么缺省值取决于matchedValuesOnly 组件，即如果matchedValuesOnly 设为TRUE，那么该位缺省为设置值；否则该位缺省为重新设置。

c) checkOverspecified 搜索控制选项是对checkOverspecified 搜索变元组件的替换。如果设置了该位，那么它对应值为TRUE 的checkOverspecified 组件。如果searchControlOptions 组件不存在，那么缺省值取决于checkOverspecified 组件，即如果checkOverspecified 组件设为TRUE，那么该位缺省为设置值；否则该位缺省为重新设置。

d) performExactly 搜索控制选项指示，合适的话，在替换基本的匹配规则后，将严格按照过滤器规定的或暗指的相关匹配规则，执行一个操作。当extensibleMatch 过滤器项规定了一个不支持的匹配规则时，如果设置了该搜索控制选项，那么将拒绝search 请求。否则，过滤器项评估为 UNDEFINED。如果搜索操作在服务特定管理区域内开始其初始评估阶段，并且搜索规则中的匹配限制出现冲突，那么当且仅当设置了该搜索控制选项，搜索规则将使搜索确认失败。

e) 只有当extendedArea 组件包括一个 0 或更大值时，includeAllAreas 搜索控制选项才相关。在所有其他情况下，它将被忽略。如果值为TRUE，那么执行包含的张弛；否则如果可能的话，执

行排他的张弛(见 GB/T 16264.2—2008 中 13.6)。

f) 当用户要求不使用 DSA 提供的张弛策略时,使用noSystemRelaxation 搜索控制选项。DSA 仍将使用基本策略,除非有一个超越它的用户提供的基本策略,但将不能使用任何后续的张弛或紧缩策略。也就是说,对候选条目集,过滤器从不评估一次以上,除非由于用户提供的张弛策略。

g) dnAttribute 搜索控制选项用于指示,除了当依据条目对过滤器进行评估时所用的那些条目属性外,还使用了哪些条目的可辨别名的属性。如果设置,它将超越extensibleMatch 过滤器项中任何可能的dnAttribute 规定。它还适用于所有的过滤器项类型。

h) 只有当设置了partialNameResolution 搜索控制选项时,matchOnResidualName 搜索控制选项才相关。它用于指示,如果目录只能解析search 操作中的部分假设名(称),那么未解析 RDN 的 AVA 将被当作是经过 AND(与)运算的equality 过滤器项。这些过滤器项与搜索过滤器做 AND("与")运算,针对的是依据搜索规则的搜索评估和条目匹配。

i) entryCount 搜索控制选项指示,在超出了服务控制大小限制或管理大小限制的情况下,将在 search 结果中提供一个条目计数。entryCount 指示,返回多少条目将拥有一个未遇到过的大小限制。如果设置了subentries 服务控制选项,那么将忽略该搜索控制。

j) useSubset 搜索控制选项指示,将忽略imposedSubset 搜索规则组件(见 GB/T 16264.2—2008 中 16.10.9)。

k) separateFamilyMembers 搜索控制选项指示,家族成员将作为单独的条目而非嵌入在family-information 派生属性中返回。

l) 如果基本客体为复合属性的一个祖(条目),那么searchFamily 搜索控制选项将规定如何执行搜索。如果基本客体不是一个祖(条目),或者如果在CommonArguments 或ChainingArguments 中设置了entryOnly,那么忽略该选项。如果设置了该选项,那么依据subset 和sizeLimit 规定,只对复合条目执行该操作,并将每个家族成员当作一个单独的条目。如果未设置searchFamily 选项,那么依据subset 规定,认为复合条目是一个单个条目。

注 6:后者意味着,作为例子,如果subset 设为baseObject,并且familyGrouping 为entryOnly,那么每个单个家族成员都在搜索范围内。

JoinArguments 变元用于规定目录的额外部分,将出于以下目的对其进行搜索,即确定和访问与主要搜索相关的条目,并规定将在联合相关条目中使用的属性。虽然规定为一个 SEQUENCE,但joinArgument 变元出现的次序并不重要。

注 7:当规定joinArguments 时,认为主要搜索和每个额外搜索都将产生一系列中间结果。来自joinArgument 规定的每个中间结果集将联合主要搜索的结果,所有的联合都将在返回SearchResult 中的任何结果之前执行。各中间结果对目录用户是不可见的。

joinBaseObject 变元用于确定相对每个将要进行的额外搜索的客体条目(或可能的话为根)。joinBaseObject 可以是一个可选的名(称),并包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

domainLocalID 变元用于任选地确定一个独立的 DIT,在其中将启动对joinBaseObject 的搜索。如果不存在,那么对joinBaseObject 的搜索将在 DSA 所知的所有 DIT 中启动。

joinSubset 变元指示额外搜索是否应用于:

a) 只有joinBaseObject;

b) 只有联合基本客体的直接下级(oneLevel);

c) 联合基本客体及其所有下级(wholeSubtree)。

joinFilter 变元用于从不关注的额外搜索空间去除条目。对联合相关条目,将只考虑符合joinFilter 要求的信息。如果不规定joinFilter,那么将使用SearchArgument filter 组件中的值。如果未提供 SearchArgument 的filter 组件,那么将使用该组件的缺省值。当出现时,依据有关extendedFilter 的规

则，将对joinFilter 进行处理。

joinAttributes 变元用于规定各属性对，它们将用于联合来自主要搜索的条目和来自额外搜索的条目。如果存在一个joinAttrPair，使以下件为 TRUE，那么认为一个来自主要搜索的条目（“主要条目”）与一个来自额外搜索的条目（“额外条目”）相关：

a) 主要条目拥有一个由baseAtt 为属性类型规定的值。

b) 额外条目拥有一个由joinAtt 为属性类型规定的值。

c) 依据以下规则，主要条目中的一个属性值和额外条目中的一个属性值是相同的：

1) 如果属性类型相同，那么对该属性类型使用等同的匹配规则。

2) 如果属性类型不相同，但有相同的语法，那么对为主要条目规定的属性类型使用等同的匹配规则。

3) 如果joinContexts 出现，那么依据上面规则 1)或规则 2)，在评估中只能使用规定上下文的属性值。如果joinContexts 不存在，那么依据上面规则 1)或规则 2)，在评估中可以使用所有上下文的属性值。

在为潜在的联合评估joinAttributes 中，将忽略联合属性的子类型。只有明确确定的baseAtt 和joinAtt 才会用于评估一个潜在的联合。

如果应用一个等同规则，并评估为 FALSE 或 UNDEFINED，那么不认为各条目是相关的。

如果在上述件 c)下没有合适的匹配规则可用，那么不认为各条目是相关的。

注 8：当规定涉及多值属性的联合时，应注意防止无意地搜索没有意义的数据。例如，如果条目使用一个多值属性，如雇员标识符，来表示委员会中的成员资格，那么在执行联合中该多值属性的规定将返回一个包含家族成员名（称）、电话号码、电子邮件等的无关联集。不过，当规定外部联合时，将返回所有的被检索条目，即使它不相关。

joinSelection 变元用于从不关注的额外搜索中间结果中去除属性。

joinType 变元用于规定将对相关条目执行的联合类型，如下所述：

a) 如果规定了innerJoin，那么结果条目集将只包括那些执行了联合的条目，它基于joinAttributes 中规定的属性对。每个结果条目都将包括所有对应的相关条目，作为related Entry属性值。

b) 如果规定了leftOuterJoin，那么结果条目集将包括所有由主要搜索选择的条目；所有执行了联合的条目（基于joinAttributes 中规定的属性对）都将包括所有对应的相关条目，作为related Entry属性值。

c) 如果规定了fullOuterJoin，那么结果条目集将包括所有来自主要搜索和额外搜索的条目；所有执行了联合的条目（基于joinAttributes 中规定的属性对）都将包括所有对应的相关条目，作为relatedEntry 属性值，而不是作为明确的条目。

除非joinAttributes 值包含至少一个JoinAttPair，并且依据匹配规则，每个JoinAttPair 都是有效的，否则不得尝试任何联合。如果不是这种情况，那么不得尝试任何联合，并且将以下作为合并各JoinAttPair 的结果，依据的是联合类型：

联合类型	合并后的输出
inner-join	空
left-outer-join	只有主要结果
full-outer-join	来自主要搜索和联合搜索的结果

否则，只有当提供所有的相关联合属性值时，条目才适于联合。

联合结果将包括匹配的联合属性的所有组合。

注 9：例如，考虑 A、B、C（作为来自主要搜索的条目）、P、Q、R（作为来自使用 J 的额外搜索的条目）、对应的JoinAttPair 值，并假设发生以下匹配是 J 的结果：

- A 与 P、A 与 Q、A 与 R；
- B 与 Q；
- C 与 P 以及 C 与 Q。

而后联合的结果将包括：

- A 与{P,Q,R}；
- B 与{Q}；
- C 与{P,Q}。

即使 Q 的结果出现三次。

CommonArguments（见 7.3）包括适用于请求的服务控制规定和安全参数。如果请求方对该操作的变元进行标记、加密或者标记和加密，那么将在变元中包括SecurityParameters（见 7.10）组件。

10.2.3 搜索结果

依据访问控制，如果找到了baseObject，而不管是否返回下级，并且如果在服务特定的管理区域内没有规定任何阻止搜索操作继续进行的服务限制，那么请求成功。

注 1：作为其必然结果，对查询同一条目属性集的读操作而言，适用于单个条目的未过滤搜索的结果可以不相同。这是因为如果在条目中不存在任何选定的属性，那么后者将返回一个AttributeError。

name 为条目的可辨别名或条目的别名，如 7.7 所述。只有当别名解除引用时、当 RDN 已解析为主要 RDN 时，或者当内容选择已应用时、当将要返回的名（称）不同于操作变元中所提供的baseObject 名（称）时，它才出现。

entries 参数从各个（0 个或多个）满足过滤器要求的条目传送请求的信息（见 7.5）。作为entries 一部分提供的名（称）可能会受 7.7 中为Name 所述的上下文影响。条目信息可以包括如EntryInformationSelection familyReturn 元素要求的家族信息。familyGrouping 与familyReturn 之间的交互作用在过滤器的四阶段评估中以及返回内容的后续评估中进行定义，如 7.8.3 所述。

partialOutcomeQualifier 如 10.1.3 所述。

注 2：如果某个特定条目的返回条目信息不完整，那么它通过返回条目信息中的incompleteEntry 参数来指出。

altMatching 用于指示未按search 请求中规定的要求准确应用匹配规则。

CommonResults notifications 元素中的appliedRelaxation 属性用于列出已放宽或收紧的过滤器属性，而不是放宽策略basic 元素提出的那些属性（见 GB/T 16264.6—2008 中 5.12.16）。

所描述的uncorrelatedSearchInfo 参数针对的是 10.1.3 中的uncorrelatedListInfo。

CommonResults（见 7.4）包括适用于响应的安全参数。如果目录对该结果进行标记、加密或者标记和加密，那么将在结果包括SecurityParameters（见 7.10）组件。

10.2.4 服务管理

管理权威部门可以建立服务特定的管理区域，如 GB/T 16264.2—2008 中第 7 章所述。这使得管理权威部门能够通过限制搜索操作来对服务进行管理，它通过定义搜索规则来限定可以搜索的 DIT 区域、可以形成的搜索类型、可以返回的信息等。

10.2.5 搜索差错

如果请求失败，那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。当在服务特定的管理区域内执行搜索时，可以返回若干额外的、非常详细的差错信息元素，详细内容见第 13 章。

10.2.6 基本访问控制的搜索操作决策点

如果rule-based-access-control 也应用，那么有关basic-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的DiscloseOnError 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对将要搜索的那部分 DIT 起作用，那么应用以下访问控制序列：

a) 对由baseObject 变元确定的条目不需要任何特殊的许可。

注 1：如果baseObject 处于SearchArgument 范围内(即当subset 变元规定baseObject 或wholeSubtree 时)，那么应用 b)～e)中规定的访问控制。

b) 对在SearchArgument 范围内的每个条目(将作为候选考虑客体)，需要浏览许可。将忽略未赋予该许可的各条目。

c) filter 变元适用于每个留待考虑 2)后才考虑的条目，依据的是以下内容：

1) 对每个规定一个属性的FilterItem，在FilterItem 被评估为 TRUE 或 FALSE 之前，对属性类型需要*FilterMatch* 许可。未赋予该许可的FilterItem 评估为 UNDEFINED。

2) 对每个额外规定一个属性值的FilterItem，对每个保存的属性值(将考虑把它用于匹配条目的)都需要*FilterMatch* 许可。如果存在一个匹配FilterItem 的值，并且赋予了许可，那么FilterItem 评估为 TRUE，否则评估为 FALSE。

d) 如果出现，那么joinCriteria 变元适用于每个留待考虑 3)后才考虑的条目，依据的是以下内容：

1) 对每个规定一个属性的JoinCriteriaItem，在JoinCriteriaItem 被评估为 TRUE 或 FALSE 之前，对属性类型需要*FilterMatch* 许可。未赋予该许可的JoinCriteriaItem 评估为 UNDEFINED。

2) 对每个额外规定一个属性值的JoinCriteriaItem，对每个保存的属性值(将考虑把它用于匹配条目的)都需要*FilterMatch* 许可。如果存在一个匹配JoinCriteriaItem 的值，并且赋予了许可，那么JoinCriteriaItem 评估为 TRUE，否则评估为 FALSE。

e) 一旦应用了在 b)～d)中定义的规程，那么要么选择条目，要么抛弃条目。如果作为对整个范围内的子树应用这些控制的结果，没有选择任何条目(partialOutcomeQualifier 中的任何ContinuationReferences 除外)，并且如果未将*DiscloseOnError* 许可赋予给由baseObject 变元确定的条目，那么操作失败，并将返回一个带问题noSuchObject 的nameError。matched 元素将包含下一个上级条目的名(称)，对该条目赋予了*DiscloseOnError* 许可，或者将包含 DIT 根的名(称)(即一个空RDNSequence)。否则，操作成功，但它不传送任何下级信息。

注 2：在返回nameError 的情况下，不访问所有上级条目的 DSA 可以使用空RDNSequence。

注 3：安全策略可以防止泄露知识信息，否则将作为partialOutcomeQualifier 中的ContinuationReferences 予以传送。如果这样一个策略发挥作用，并且如果 DUA 通过规定chainingProhibited 来约束服务，那么目录可能返回一个带问题chainingRequired 的serviceError。否则，将从partialOutcomeQualifier 中省略ContinuationReference。

f) 否则，对每个选定的条目，返回的信息如下所述：

1) 如果selection 的infoTypes 元素规定只能返回属性类型，那么对每个将要返回的属性类型，需要读许可。如果未赋予许可，那么从EntryInformation 中删去属性类型。如果作为应用这些控制的结果，没有选择任何属性类型信息，那么返回EntryInformation 元素，但不用它传送任何属性类型信息(即省略SET OF CHOICE 元素或为空)。

2) 如果selection 的infoTypes 元素规定返回属性类型和值，那么对每个将要返回的属性类型和值，需要读许可。如果对属性类型未赋予许可，那么从EntryInformation 中删去属性。如果对属性值未赋予许可，那么从其对应的属性中删去值。在未将许可赋予属性内任何值的情况下，返回一个包含空SET OF AttributeValue 的Attribute 元素。如果作为应用这些控制的结果，没有选择任何属性信息，那么返回EntryInformation 元素，但不用它传送任何属性信息(即省略SET OF CHOICE 元素或为空)。

注 4：如果*DiscloseOnError* 许可未赋予baseObject 变元确定的条目，那么不应返回指出limitProblem 或unavailableCriticalExtensions 的partialOutcomeQualifier，原因是它可能对该条目的安全造成危害。

10.2.6.1 在额外搜索情况下基本访问控制的搜索操作决策点

如果joinArguments 变元出现，并且如果basic-access-control 对将要搜索的那部分 DIT 起作用，那么对每个附加搜索应用以下访问控制序列：

a) 对由joinBaseObject 变元确定的条目不需要任何特殊的许可。

注 1：如果joinBaseObject 处于joinArgument 范围内(即当joinSubset 变元规定baseObject 或wholeSubtree 时)，那么应用 b)～ f)中规定的访问控制。

b) 对在joinArgument 范围内的每个目(将作为候选考虑客体)，需要浏览许可。将忽略未赋予该许可的各条目。

c) 如果出现，那么joinFilter 变元适用于每个留待考虑 b)后才考虑的条目，依据的是以下内容：

1) 对每个规定一个属性的FilterItem，在FilterItem 被评估为 TRUE 或 FALSE 之前，对属性类型需要*FilterMatch* 许可。未赋予该许可的FilterItem 评估为 UNDEFINED。

2) 对每个额外规定一个属性值的FilterItem，对每个保存的属性值(将考虑把它用于匹配条目的)都需要*FilterMatch* 许可。如果存在一个匹配FilterItem 的值，并且赋予了许可，那么FilterItem 评估为 TRUE，否则评估为 FALSE。

d) 如果joinFilter 变元不出现，那么filter 变元适用于每个留待考虑 b)后才考虑的条目，依据的是以下内容：

1) 对每个规定一个属性的FilterItem，在FilterItem 被评估为 TRUE 或 FALSE 之前，对属性类型需要*FilterMatch* 许可。未赋予该许可的FilterItem 评估为 UNDEFINED。

2) 对每个额外规定一个属性值的FilterItem，对每个保存的属性值(将考虑把它用于匹配条目的)都需要*FilterMatch* 许可。如果存在一个匹配FilterItem 的值，并且赋予了许可，那么FilterItem 评估为 TRUE，否则评估为 FALSE。

e) 一旦应用了在 b)～d)中定义的规程，那么要么选择条目，要么抛弃条目。如果作为对整个范围内的子树应用这些控制的结果，没有选择任何条目(partialOutcomeQualifier 中的任何ContinuationReferences 除外)，并且如果未将*DiscloseOnError* 许可赋予给由aseObject 变元确定的条目，那么操作失败，并将返回一个带问题noSuchObject 的nameError。matched 元素将包含下一个上级条目的名(称)，对该条目赋予了*DiscloseOnError* 许可，或者将包含 DIT 根的名(称)(即一个空RDNSequence)。否则，操作成功，但它不传送任何下级信息。

注 2：在返回nameError 的情况下，不访问所有上级条目的 DSA 可以使用空RDNSequence。

注 3：安全策略可以防止泄露知识信息，否则将作为partialOutcomeQualifier 中的ContinuationReferences 予以传送。如果这样一个策略发挥作用，并且如果 DUA 通过规定chainingProhibited 来约束服务，那么目录可能返回一个带问题chainingRequired 的serviceError。否则，将从partialOutcomeQualifier 中省略ContinuationReference。

f) 否则，对每个选定的条目，返回的信息如下所述：

1) 如果selection 的infoTypes 元素规定只能返回属性类型，那么对每个将要返回的属性类型，需要读许可。如果未赋予许可，那么从EntryInformation 中删去属性类型。如果作为应用这些控制的结果，没有选择任何属性类型信息，那么返回EntryInformation 元素，但不用它传送任何属性类型信息(即省略SET OF CHOICE 元素或为空)。

2) 如果selection 的infoTypes 元素规定返回属性类型和值，那么对每个将要返回的属性类型和值，需要读许可。如果对属性类型未赋予许可，那么从EntryInformation 中删去属性。如果对属性值未赋予许可，那么从其对应的属性中删去值。在未将许可赋予属性内任何值的情况下，返回一个包含空SET OF AttributeValue 的Attribute 元素。如果作为应用这些控制的结果，没有选择任何属性信息，那么返回EntryInformation 元素，但不用它传送任何属性信息(即省略SET OF CHOICE 元素或为空)。

注 4：如果*DiscloseOnError* 许可未赋予baseObject 变元确定的条目，那么不应返回指出limitProblem 或

unavailableCriticalExtensions 的partialOutcomeQualifier，原因是它可能对该条目的安全造成危害。

10.2.6.2 搜索期间解除引用别名

对search 操作过程中发生的别名解除引用不需要任何特殊的许可（由于searchAliases 参数设为TRUE）。不过，对每个遇到的别名条目，如果别名解除引用将导致在partialOutcomeQualifier 中返回ContinuationReference，那么将应用以下访问控制：对别名条目、aliasedEntryName 属性及其包含的单个值需要读许可。如果未赋予任何这些许可，那么将从partialOutcomeQualifier 中删去ContinuationReference。这些访问控制也适用于在另一个 DSA 响应中收到的continuationReference。也就是说，DSA 将管辖所有的continuationReferences，不论它们是否在本地产生。

注：除了上面所述的访问控制，安全策略可以防止泄露知识信息，否则将作为partialOutcomeQualifier 中的ContinuationReferences 予以传送。如果这样一个策略发挥作用，并且如果 DUA 通过规定chainingProhibited 来约束服务，那么目录可能返回一个带问题chainingRequired 的serviceError。否则，将从partialOutcomeQualifier 中省略ContinuationReference。

10.2.6.3 不泄露不完整的结果

如果在EntryInformation 中返回一个不完整的结果，即因适用的访问控制而删去了某些属性或属性值，如果*DiscloseOnError* 许可赋给至少一个结果中保留的属性类型，或者赋给至少一个结果中保留的属性值（对该属性类型赋予了*读*许可），那么incompleteEntry 元素将被设为 TRUE。

10.2.7 基于规则的访问控制的搜索操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-contro 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在执行search 操作的那部分 DIB 起作用，那么应用以下访问控制序列：

a) 如果对由baseObject 变元确定的条目拒绝基于规则的条目级许可，那么返回带问题noSuchObject 的nameError，如 7.11.2.4 定义；

b) 在rule-based-access-control 下，将忽略SearchArgument 范围内的每个条目，对SearchArgument，拒绝条目级访问；

c) 应用有关条目的basic-access-control，如 10.2.6 的 b）定义；

d) 应用filter，忽略在rule-based-access-control 下拒绝访问的属性值；

e) 应用有关filter 的basic-access-control，如 10.2.6 的 c）和 d）定义；

f) 对任何选定的条目：

1) 对每个在rule-based-access-control 下可能返回的属性类型，务必将访问赋予该类型的至少一个属性值；

2) 将不返回在rule-based-access-control 下拒绝访问的属性值；

g) 对返回的信息应用basic-access-control，如 10.2.6 的 e）定义。

11 目录修改操作

有四种操作可用于修改目录：分别是在 11.1 至 11.4 中定义的addEntry、removeEntry、modifyEntry 和modifyDN。

注 1：这些操作中的每一个都通过其可辨别名来确定目标条目。

注 2：addEntry、removeEntry 和modifyDN 操作的成功执行可能依赖于跨越目录的 DIB 的物理分布。如果失败，将报告带问题affectsMultipleDSAs 的updateError。见 GB/T 16264.4—2008。

注 3：在基本通信机制发生故障的情况下，操作的结果是不确定的。用户应使用目录查询操作来检查尝试的修改操作是否取得了成功。

11.1 增加条目

11.1.1 增加条目语法

addEntry 操作用于向 DIT 增加一个叶条目(一个客体条目或一个别名条目)。操作变元可以由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 的中 17.3)。如果这样请求,那么目录可以对结果进行标记、加密或者标记和加密。

```
addEntry OPERATION ::={
        ARGUMENT            AddEntryArgument
        RESULT              AddEntryResult
        ERRORS              {attributeError|nameError|serviceError|referral|securityError|
                            updateError }
        CODE                id -opcode-addEntry }
AddEntryArgument ::=OPTIONALLY-PROTECTED {
        SET {
                object                  [0]  Name,
                entry                   [1]  SET OF Attribute,
                targetSystem            [2]  AccessPoint OPTIONAL,
                COMPONENTS OF           CommonArguments} }
AddEntryResult ::=CHOICE {
        null            NULL,
        information     OPTIONALLY-PROTECTED-SEQ {
                        SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.1.2 增加条目变元

object 变元用于确定将要增加的条目。其直接上级(为了操作取得成功,它必须已经存在)通过移除最后一个 RDN 组件(它属于将要创建的条目)来确定。object 可以是一个可选的名(称),并可以包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。最后一个 RDN 组件将是主要的 RDN,并将包括所有的不同值,其所有属性的上下文清单对 RDN 起作用。如果最后一个 RDN 组件中提供的任何 AttributeTypeAndDistinguishedValue 都不带可选的不同值,那么提供的单个值将用作该属性的单个不同值。

entry 变元包含属性信息,与来自 RDN 的信息一起组成将要创建的条目。目录将确保条目复合目录方案要求。如果正在创建的条目是一个别名,那么不需要任何检查即可确保aliasedEntryName 属性指向一个有效的条目。

targetSystem 变元指示 DSA 持有新的条目。如果该变元不存在,那么它将意味着同一 DSA 持有新客体的上级。如果该变元出现,那么它将是带AccessPoint 的 DSA。当增加子条目时,参数不存在。

如果变元存在,那么将对CommonArguments 中criticalExtensions 参数中的targetSystem 位进行设置,指示该扩展是重要的。

注 1:如果指示或暗指 DSA 的选择与本地管理策略冲突,那么不执行操作并返回一个差错。

CommonArguments(见 7.3)包括有关服务控制和适用于请求的安全参数的规定。除非在critical Extensions中设置useAliasOnUpdate 临界扩展位,否则忽略dontDereferenceAlias 选项(并当作已经设置了)。因此,只有当不设置dontDereferenceAlias 并且设置useAliasOnUpdate 时,才通过该操作解除引用别名。如果提供,那么忽略sizeLimit 组件。如果请求方对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters(见 7.10)组件将包括在变元中。

注 2:如果遇到第 1 版本 DSA,那么涉及解除引用别名的更新操作将总失败。

11.1.3 增加条目结果

如果请求成功,那么将返回一个结果。如果目录对该结果进行标记、加密或者标记和加密,那么

CommonResultsSeq(见 7.4)的SecurityParameters 组件(见 7.10)将包括在结果中。如果目录不对该操作的结果进行标记,那么不随结果传送任何信息。

11.1.4 增加条目差错

如果请求失败,那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。

11.1.5 基本访问控制的增加条目操作决策点

如果rule-based-access-control 也应用,那么有关basic-access-control 应用的次序问题将是一个本地问题,除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的DiscloseOnErro 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在增加的条目起作用,那么应用以下访问控制序列:

a) 对由object 变元确定的条目的直接上级不需要任何特殊的许可。

注 1:安全策略可以防止目录用户越过 DSA 边界来增加条目(例如,利用targetSystem 变元)。在这种情况下,可能返回一个适当的nameError、serviceError、securityError 或updateError,前提是它不会危害直接上级条目的存在。如果这样(即未将*DiscloseOnError* 赋予上级条目),那么之后将执行 7.11.3 中定义的、关于上级条目的规程将紧跟其后。

b) 如果条目已经存在,并且可辨别名等于object 变元,那么依据 11.1.5.1a),该操作失败。

c) 对正在增加的新条目需要增加许可。如果不赋予该许可,那么依据 11.1.5.1b),该操作失败。

注 2:当尝试增加一个条目时,增加许可将作为prescriptiveACI提供,当尝试增加一个子条目时,增加许可将作为prescriptiveACI 或subentryACI 提供。

d) 对将要增加的每个属性类型和每个值,需要增加许可。如果未出现任何许可,那么依据 11.1.5.1c),该操作失败。

11.1.5.1 差错返回

如果操作失败,如 11.1.5 定义,那么应用以下规程:

a) 如果操作失败,如 11.1.5b)定义,那么有效的差错返回为以下之一:如果DiscloseOnError 或增加许可赋予给了现有的条目,那么将返回一个带问题entryAlreadyExists 的updateError。否则对正在增加的条目,紧接着执行如 7.11.3 所述的规程。

b) 如果操作失败,如 11.1.5c)定义,那么对正在增加的条目,紧接着执行如 7.11.3 所述的规程。

c) 如果操作失败,如 11.1.5d)定义,那么有效的差错返回为带问题insufficientAccessRights 或noInformation 的securityError。

11.1.6 基于规则的访问控制的增加条目操作决策点

如果basic-access-control 也应用,那么有关rule-based-access-control 应用的次序问题将是一个本地问题,除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在执行addEntry 操作的那部分 DIB 起作用,那么应用以下访问控制序列:

a) 如果拒绝赋予直接上级基于规则的条目级许可,那么返回带问题noSuchObject 的nameError,如 7.11.2.4 定义。

b) 应用basic-access-control,如 11.1.5 定义。

11.2 移除条目

11.2.1 移除条目语法

移除条目操作用于从 DIT 移除一个叶条目(一个客体条目、家族成员或别名条目)或一个非叶祖

(条目)及其孩子。操作变元可以由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 中 17.3)。如果这样请求,那么目录可以对结果进行标记、加密或者标记和加密。

```
removeEntry OPERATION ::={
        ARGUMENT        RemoveEntryArgument
        RESULT          RemoveEntryResult
        ERRORS          { nameError|serviceError|referral|securityError|updateError }
        CODE            id-opcode-removeEntry }
RemoveEntryArgument ::=OPTIONALLY-PROTECTED {
      SET {
            object                  [0]  Name,
            COMPONENTS OF                CommonArguments } }
RemoveEntryResult ::=CHOICE {
      null                  NULL,
      information           OPTIONALLY-PROTECTED-SEQ {
          SEQUENCE { COMPONENTS OF CommonResultsSeq } } }
```

11.2.2 移除条目变元

object 变元用于确定将要删除的条目。object 可以是一个可选的名(称),并可以包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

CommonArguments(见 7.3)包括有关服务控制和适用于请求的安全参数的规定。除非在criticalExtensions中设置useAliasOnUpdate 临界扩展位,否则忽略dontDereferenceAlias 选项(并当作已经设置了)。因此,只有当不设置dontDereferenceAlias 并且设置useAliasOnUpdate 时,才通过该操作解除引用别名。如果提供,那么忽略sizeLimit 组件。如果请求方对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters(见 7.10)组件将包括在变元中。

注:如果遇到第 1 版本 DSA,那么涉及解除引用别名的更新操作将总失败。

FamilyGrouping 可以按如下设置:

- 对该操作,entryOnly 为缺省值。将要移除的条目将是一个叶条目。
- 可以为祖(条目)规定compoundEntry。将移除复合条目的所有成员。如果目标客体不是祖(条目),那么操作失败,返回带问题notAncestor 的updateError。如果不可能移除所有成员,例如出于安全原因,那么操作也将失败,返回一个适当的差错。

如果FamilyGrouping 不出现或设置为上述值之外的任何其他值,那么采用entryOnly。

11.2.3 移除条目结果

如果请求成功,那么将返回一个结果。如果目录对该结果进行标记、加密或者标记和加密,那么CommonResultsSeq(见 7.4)的SecurityParameters 组件(见 7.10)将包括在结果中。如果目录不对该操作的结果进行标记,那么不随结果传送任何信息。

当EntryInformationSelection 中的familyReturn 选择家族信息时,返回的信息在 7.6.4 中定义。

information 组件中返回的信息对应(成功)执行修改条目操作后的 DIB 状态。

11.2.4 移除条目差错

如果请求失败,那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。

11.2.5 基本访问控制的移除条目操作决策点

如果rule-based-access-control 也应用,那么有关basic-access-control 应用的次序问题将是一个本地问题,除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control

拒绝的许可。

如果basic-access-control 对正在移除的条目起作用，那么应用以下访问控制：

- 对正在移除的条目需要移除许可。如果不赋予该许可，那么依据 7.11.1，该操作失败。

注：对出现在被移除条目中的任何属性和属性值，不需要任何特殊的许可。

11.2.6 基于规则的访问控制的移除条目操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的DiscloseOnError 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在移除的条目起作用，那么应用以下访问控制序列：

a) 如果未赋予目标条目基于规则的条目级许可，那么操作失败，返回带问题noSuchObject 的nameError，如 7.11.2.4 定义；

b) 应用条目级basic-access-control，如 11.2.5 规定；

c) 如果未赋予属性值基于规则的访问，那么它将不被移除；

d) 如果未赋予基于规则的 RDN 访问，那么 RDN 的任何属性值都将不被移除。如果移除所有的属性值，那么从条目中移除属性。如果移除所有的属性，那么从 DIT 中移除条目。如果移除至少一个属性值，并且请求方没有 RDN 许可，那么操作成功，但条目继续留在 DIT 中，带一个或多个属性；

注 1：除非条目不同值的标签上下文的所有值都有相同的值，否则这不支持基于规则的访问控制策略。

e) 在rule-based-access-control 下，如果赋予了 RDN 许可，但未赋予访问至少一个其他属性值的许可，那么不移除 RDN，操作失败，返回带问题insufficientAccessRights 的securityError。是否移除请求方拥有访问许可的其他属性值是一个本地问题；

注 2：这向请求方表明，至少存在一个无法访问的属性值。

f) 如果移除条目的所有属性，那么从 DIT 中移除该条目，操作成功。

11.3 修改条目

11.3.1 修改条目语法

修改条目操作用于对单个条目执行一系列以下修改中的一个或多个：

a) 增加一个新的属性；

b) 移除一个属性；

c) 增加属性值；

d) 移除属性值；

e) 替换属性值；

f) 修改一个别名；

g) 增加一个常量或一个属性的所有值；

h) 删去所有属性值，在每个上下文中回退为 FALSE。

操作变元可以由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 中 17.3）。如果这样请求，那么目录可以对结果进行标记、加密或者标记和加密。

```
modifyEntry OPERATION ::={
        ARGUMENT        ModifyEntryArgument
        RESULT          ModifyEntryResult
        ERRORS          {attributeError|nameError|serviceError|referral|securityError|
                        updateError}
```

```
        CODE                    id-opcode-modifyEntry}
    ModifyEntryArgument ::=OPTIONALLY-PROTECTED{
        SET{
            object                  [0]  Name,
            changes                 [1]  SEQUENCE OF EntryModification,
            selection               [2]  EntrylnformationSelection OPTIONAL,
            COMPONENTS OF           CommonArguments}}
    ModifyEntryResult ::=CHOICE{
        null                        NULL,
        information                 OPTIONALLY-PROTECTED -SEQ{
            SEQUENCE {
            entry               [0]  EntrylnformationOPTIONAL,
            COMPONENTS OF       CommonResultsSeq}}}
    EntryModification ::= CHOICE{
        addAttribute        [0]  Attribute,
        removeAttribute     [1]  AttributeType,
        addValues           [2]  Attribute,
        removeValues        [3]  Attribute,
        alterValues         [4]  AttributeTypeAndValue,
        resetValue          [5]  AttributeType,
        replaceValues       [6]  Attribute}
```

11.3.2 修改条目变元

object 变元用于确定适用于修改的条目。object 可以是一个可选的名(称),并可以包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

changes 变元用于确定修改序列,它以规定的次序应用。如果任何一个单个修改失败,那么产生一个attributeError,条目仍处于操作之前它的状态。也就是说,操作是很小的。修改序列的最终结果不应与目录方案出现冲突。不过,对单个EntryModification 修改它可能这样做,并且有时候需要这样做。可能发生以下类型的修改:

a) addAttribute:它确定一个将要加入条目的新属性,它完全由变元确定。任何增加一个已经存在的属性的尝试都将导致一个attributeError。

b) removeAttribute:变元用于确定(通过其类型)一个将要从条目中移除的属性。任何移除一个不存在的属性的尝试都将导致一个attributeError。

   注 1:如果属性值出现在 RDN 中,那么将不允许进行本操作。

c) addValues ——通过变元中的变元类型,它确定一个属性,并规定一个或多个将要加入属性的属性值。任何增加一个已经存在的值的尝试都将导致一个差错。任何向一个不存在的类型增加一个值的尝试都将导致类型和值的增加。

d) removeValues ——通过变元中的变元类型,它确定一个属性,并规定一个或多个将要从属性中移除的属性值。如果值未出现在属性中,那么将导致一个attributeError。任何从一个属性中移除最后一个值的尝试都将导致移除属性类型。

   注 2:如果其中之一的值出现在 RDN 中,那么将不允许进行本操作。

   可以通过或不通过一个上下文清单来规定将要增加的属性或属性值。上下文不能加入现有的属性值,不能从现有的属性值中移除,也不能修改。为了更改现有属性值的上下文清单,首先需要移除属性值,而后以新的上下文清单插入相同的属性值。当移除一个属性值时,将不

提供任何上下文清单，并且与正要移除的属性值关联的任何现有上下文清单都将随属性值移除。

e) alterValues：它确定一个属性类型，并规定一个将要加入所有属性值的数量。尝试对语法不是为INTEGER 或REAL 的属性进行修改将导致一个attributeError。

f) resetValue：它通过其类型确定一个属性，并移除属性的所有值（如果有的话），它有一个关联的属性值上下文，其退路为FALSE。resetValue 不移除任何没有上下文的属性值。

g) replaceValues：它用提供的值替换给定属性类型的所有现有值，如果它不存在，那么创建属性类型。如果它存在，不带值的替换将移除属性类型，如果类型不存在，将忽略之。

注 3：本目录规范不建立有关执行 DSA 对其所接收的 PDU 进行解码和处理的次序的规则。如果在处理每个元素之前，DSA 对整个 PDU 进行解码，并且如果对非可选的选项存在一个新的和非预期的值，如replaceValues，那么 DSA 将发出一个编码差错信号。不过，如果 DSA 根据需要对各元素进行解码，那么它很可能将检测到一个未知的临界扩展，并返回一个不支持的临界扩展理由代码，告知操作失败。在任何一种情况下，DSA 不对操作进行处理都是正确的；不过，执行方应意识到，任何一种信号都可以用来指示操作失败。

变元将被单个ModifyEntry 操作中的addValues 和removeValues 组合所替代。

CommonArguments（见 7.3）包括有关服务控制和适用于请求的安全参数的规定。除非在criticalExtensions 中设置useAliasOnUpdate 临界扩展位，否则忽略dontDereferenceAlias 选项（并当作已经设置了）。因此，只有当不设置dontDereferenceAlias 并且设置useAliasOnUpdate 时，才通过该操作解除引用别名。如果提供，那么忽略sizeLimit 组件。如果请求方对该操作的变元进行标记、加密或者标记和加密，那么SecurityParameters（见 7.10）组件将包括在变元中。

注 4：如果遇到第 1 版本 DSA，那么涉及解除引用别名的更新操作将总失败。

selection 变元用于规定一个可选的条目信息选择，它用于控制是否在操作结果中返回信息，并规定返回的特定属性和值。只有当通过绑定操作商定的版本为v2 或更高时，才规定它。

操作可用于修改目录操作属性。只能对那些不是分类noUserModification（并且用户拥有有效的修改访问权限）的目录操作属性进行修改。

注 5：无论用户修改是否允许，目录都可以对目录操作属性的值进行修改，作为其他目录操作的副作用。

只有当服务控制subentries 为TRUE，并且object 为实际持有待修改之联合属性的分条目时，操作才可用于修改联合属性。

注 6：因此，在修改读条目返回的信息时需小心：某些信息可能来自属性集，在针对条目本身的操作中不能对之进行修改。例如，不可能通过针对条目的removeAttribute 条目修改来从一个（普通）条目中删去属性集（将返回一个带问题noSuchAttributeOrValue 的attributeError）。

如果值规定了辅助客体类，那么操作可用于修改一个条目的客体类属性值。不过，如果尝试修改一个客体类值（它规定了一个条目的结构客体类），那么将导致一个带问题objectClassModificationProhibited 的updateError。对辅助客体类的任何修改将使超类链接与作为结果的客体类定义保持一致与正确。

11.3.3 修改条目结果

如果请求成功，那么将返回一个result。如果在操作变元中没有规定任何selection，并且不对结果进行标记、加密或者标记和加密，那么返回空结果。如果没有规定任何selection（但目录将对结果进行标记、加密或者标记和加密），那么省略条目组件。如果目录对结果进行标记、加密或者标记和加密，那么CommonResultsSeq（见 7.4）的SecurityParameters 组件（见 7.10）将包括在结果中。如果目录不对结果进行标记，那么不随结果传送任何条目信息。

11.3.4 修改条目差错

如果请求失败，那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。

11.3.5 基本访问控制的修改条目操作决策点

如果rule-based-access-control 也应用，那么有关basic-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在修改的条目起作用，那么应用以下访问控制序列：

a) 对正在修改的条目需要修改许可。如果不赋予该许可，那么依据 7.11.1，该操作失败。

b) 对在序列中应用的、每个规定的 EntryModification 变元，需要以下许可：

1) 对在addAttribute 参数中规定的属性类型和每个值的增加许可。如果不赋予这些许可或属性已经存在，那么依据 11.3.5.1a)，该操作失败。

2) 对在removeAttribute 参数中规定的属性类型的删除许可。如果不赋予该许可，那么依据 11.3.5.1b)，该操作失败。

注 1：对出现在被移除属性中的任何属性值，不需要任何特殊的许可。

3) 对在 addValues 参数中规定的每个属性值的增加许可。如果不赋予这些许可或任何属性值已经存在，那么依据 11.3.5.1c)，该操作失败。

4) 对在removeValues 参数中规定的每个值的移除许可。如果不赋予这些许可，那么依据 11.3.5.1d)，该操作失败。

注 2：如果removeValues 修改的最终结果是移除属性的最后一个值(它将移除属性自身)，那么对规定的属性类型也需要移除许可。

5) 对在alterValues 参数中规定的每个值的增加和移除许可。如果不赋予这些许可，那么依据 11.3.5.1e)，该操作失败。

6) 对将要通过resetValue 参数移除的每个值的移除许可。如果将至少移除一个值并且不赋予这些许可，那么依据 11.3.5.1f)，该操作失败。

11.3.5.1 差错返回

如果操作失败，如 11.3.5 定义，那么应用以下规程：

a) 如果操作失败，如 11.3.5 的 b)的 1)所定义，那么有效的差错返回是以下之一：如果属性已经存在，并且将*DiscloseOnError* 或增加赋予了该属性，那么将返回一个带问题attributeOrValueAlreadyExists 的attributeError；否则，将返回一个带问题insufficientAccessRights 或noInformation 的securityError。

b) 如果操作失败，如 11.3.5 的 b)的 2)所定义，那么有效的差错返回是以下之一：如果将DiscloseOnError 许可赋予了正要移除的属性，并且该属性存在，那么将返回一个带问题insufficientAccessRights 或 noInformation 的 securityError；否则，将返回一个带问题 noSuchAttributeOrValue 的attributeError。

c) 如果操作失败，如 11.3.5 的 b)的 3)所定义，那么有效的差错返回是以下之一：如果属性值已经存在，并且将DiscloseOnError 或增加赋予了该属性值，那么将返回一个带问题attributeOrValueAlreadyExists 的attributeError；否则，将在属性级上对DiscloseOnError 许可进行验证。如果将DiscloseOnError 赋予了该属性，那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError；否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

d) 如果操作失败，如 11.3.5 的 b)的 4)所定义，那么有效的差错返回是以下之一：如果将DiscloseOnError 许可赋予了任何正要移除的属性值，那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError；否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

e) 如果操作失败，如 11.3.5 的 b)的 5)所定义，那么有效的差错返回是以下之一：如果将DiscloseOnError 许可赋予了任何正要更改的属性值，那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError；否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

f) 如果操作失败，如 11.3.5 的 b)的 6)所定义，那么有效的差错返回是以下之一：如果将DiscloseOnError 许可赋予了任何正要移除的属性值，那么将返回一个带问题insufficientAccessRights 或noInformation 的securityError；否则，将返回一个带问题noSuchAttributeOrValue 的attributeError。

11.3.6 基于规则的访问控制的修改条目操作决策点

如果basic-access-control 也应用，那么有关rule-based-access-control 应用的次序问题将是一个本地问题，除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问，那么它将不被其他机制超越。在这种情况下，basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在修改的条目起作用，那么应用以下访问控制序列：

a) 如果未赋予目标条目基于规则的条目级许可，那么依据 7.11.2.4，操作失败，返回带问题noSuchObject 的 nameError。

b) 使用 11.3.5.1 中条目级basic-access-control。

c) 务必准予对每个移除的属性值(如果有的话)进行访问。如果不将rule-based-access-control 许可赋予任何将要移除的属性值，那么操作失败，返回带问题noSuchAttributeOrValue 的attributeError。

d) 使用 11.3.5 的 b)中属性级basic-access-control。

11.4 修改 DN

11.4.1 修改 DN 语法

修改 DN 操作用于修改条目的相对可辨别名、修改条目的主要相对可辨别名、增加和减少属性的不同值，与/或将条目移至 DIT 的一个新上级。它可以与客体条目一起使用，包括复合条目或别名条目。

对家族成员，其使用限于以下情况，即受影响的家族成员仍将留在同一复合条目中。

如果条目有下级，那么相应地对所有下级进行重新命名或移动(即子树仍保持完整)。请求方可以对操作变元进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 中 17.3)。如果这样请求，那么目录可以对结果进行标记、加密或者标记和加密。

注 1：第 1 版本系统只能将该操作用于修改叶条目的相对可辨别名。

注 2：只有当旧的上级、新的上级、条目及其所有下级都在一个 DSA 中时，第 2 版本和后续版本系统才能使用该操作将条目移至一个新的上级。

注 3：操作不将条目移至一个新的上级；所有的条目都将继续留在最初的 DSA 中。

注 4：整体上操作成功或失败；某些条目移动、某些条目不移动将不算失败。对目录的用户，操作的任何中间状态都将是外部不可见的。

注 5：在该操作后可能需要某些离线行为，以便保持一致性，例如对任何持有可辨别名值的条目的属性进行更新，指的是重新命名或移除的各条目。

注 6：对重新命名或移除条目的下级条目，不对其modifyTimeStamp 属性进行更新。

```
modifyDN OPERATION ::= {
    ARGUMENT        ModifyDNArgument
    RESULT          ModifyDNResult
    ERRORS          { nameError | serviceError | referral | securityError | updateError }
    CODE            id-opcode-modifyDN }
```

```
ModifyDNArgument ::= OPTIONALLY-PROTECTED {
    SET {
        object          [0]  DistinguishedName,
        newRDN          [1]  RelativeDistinguishedName,
        deleteOIdRDN    [2]  BOOLEAN DEFAULT FALSE,
        newSuperior     [3]  DistinguishedName OPTIONAL,
        COMPONENTS OF   CommonArguments } }
ModifyDNResult ::= CHOICE{
    null            NULL,
    information     OPTIONALLY-PROTECTED-SEQ {
        SEQUENCE {
            newRDN              RelativeDistinguishedName,
            COMPONENTS OF       CommonResultsSeq } } }
```

11.4.2 修改 DN 变元

object 变元用于确定将对其可辨别名进行修改的条目。将不解除引用名(称)中的各别名。object 可以是一个可选的名(称),并可包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

newRDN 变元用于规定条目新的 RDN。如果操作将条目移至一个新的上级,而不改变其 RDN,那么为该参数提供旧的 RDN。

如果在条目中尚未存在新的 RDN 属性值(作为旧的 RDN 的一部分或者作为非不同值),那么增加之。如果不能增加,那么返回一个差错。

对每个对 RDN 起作用的属性,如果不同的值通过上下文区分,那么newRDN 可以提供可选的不同值,如 GB/T 16264.2—2008 中 9.3 所述。如果这样,那么newRDN 将是一个主要RDN,将包括所有的不同值,其针对所有属性的上下文清单对 RDN 起作用(包括继续保留作为不同值的现有的不同值)。不带可选不同值而提供的、newRDN 中的一个AttributeTypeAndDistinguishedValue 用于指示有关该属性的一个单个不同值。

如果设置了deleteOldRDN 标志,那么不在新的 RDN 中的旧的 RDN 中的所有属性值都将被删去。这包括通过上下文区分的可选不同值,如果它们存在于旧的 RDN 中而未包括在新的 RDN 中。如果未设置该标志,那么旧的不同值仍将留在条目中(但不再是不同值)。当通过操作改变了 RDN 中的一个单个值属性,那么将设置该标志。如果旧的 RDN 中的属性值等同于新的 RDN 中的属性值(除了其上下文清单),那么旧的 RDN 中的属性值将被新的 RDN 中的属性值所替换。如果该操作移除了属性的最后一个属性值,那么该属性将被删去。

如果变元存在,那么newSuperior 规定条目将被移至 DIT 中的一个新的上级。条目变成为带指示可辨别名的条目的一个直接下级,它必须是一个已经存在的客体条目。新的上级不会是条目自身,或者任何其属,或者一个别名,否则移除的条目将与任何 DIT 结构规则都将产生冲突。移除条目的条目下级有可能与活动的子方案产生冲突,在这种情况下,子方案管理权威部门负责对这些条目进行后续调整,使之与子方案保持一致,如 GB/T 16264.2—2008 中第 14 章所述。

如果变元存在,那么将对CommonArguments 中criticalExtensions 参数中的newSuperior 位进行设置,指示该扩展是重要的。

newSuperior 可以是一个可选的名(称),并可以包括上下文信息,如 GB/T 16264.2—2008 中 9.3 所述。

CommonArguments(见 7.3)包括有关服务控制和适用于请求的安全参数的规定。出于该操作的目的,dontDereferenceAlias 选项与sizeLimit 组件不相关,如果提供,将被忽略。该操作从不解除引用别名。如果请求方对该操作的变元进行标记、加密或者标记和加密,那么SecurityParameters(见 7.10)

组件将包括在变元中。

11.4.3 修改 DN 结果

如果请求成功,那么将返回一个结果。如果目录对该结果进行标记、加密或者标记和加密,那么CommonResultsSeq(见 7.4)的SecurityParameters 组件(见 7.10)以及新的 RDN 将包括在结果中。如果目录不对结果进行标记,那么不随结果传送任何信息。

11.4.4 修改 DN 差错

如果请求失败,那么将报告其中一个列出的差错。对将报告特殊差错的情况在第 12 章中进行定义。

11.4.5 基本访问控制的修改 DN 操作

如果rule-based-access-control 也应用,那么有关basic-access-control 应用的次序问题将是一个本地问题,除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果basic-access-control 对正在重新命名的条目起作用,那么应用以下访问控制:

- 如果操作的作用是修改条目的 RDN,那么对正在重新命名的条目需要重新命名许可(考虑其最初的名(称))。如果不赋予该许可,那么依据 11.4.5.1,操作失败。
- 如果操作的作用是将一个条目移至 DIT 的一个新上级,那么对正在考虑其最初名(称)的条目需要输出许可,对正在考虑其新名(称)的条目需要输入许可。如果不赋予这些许可中的任何一个许可,那么依据 11.4.5.1,操作失败。

注 1:输入许可应作为说明性 ACI 提供。

注 2:无需任何额外的许可,即使作为修改名(称)最后 RDN 的结果,需要增加一个新的不同值或删除一个旧的值。

11.4.5.1 差错返回

如果操作失败,如 11.4.5 所定义,那么紧跟着将执行 7.11.1 中所述的规程,它有关重新命名的条目(考虑其最初的名(称))。

11.4.6 基于规则的访问控制的修改 DN 操作

如果basic-access-control 也应用,那么有关rule-based-access-control 应用的次序问题将是一个本地问题,除非如果任何一个机制都拒绝对条目、属性类型或属性值的访问,那么它将不被其他机制超越。在这种情况下,basic-access-control 的*DiscloseOnError* 许可是一种将不超越rule-based-access-control 拒绝的许可。

如果rule-based-access-control、rule-and-basic-access-control 或rule-and-simple-access-control 对正在重新命名的条目起作用,那么应用以下访问控制序列:

a) 如果未赋予目标条目基于规则的 RDN 许可,那么依据 7.11.2.4,操作失败,返回带问题noSuchObject 的 nameError。

b) 依据 11.4.5,应用条目级basic-access-control。

c) 如果操作的作用是将条目移至 DIT 的一个新上级,那么对新的上级需要基于规则的 RDN 许可,否则依据 7.11.2.4,操作失败,返回带问题noSuchObject 的nameError。

12 差错

12.1 差错优先级

目录不继续执行以下点之外的操作,即它在该点确定报告差错。

注 1:该规则的一个含义是,对相同查询的重复例子,遇到的第一个差错可以不同,原因是,对处理某个给定的查询,没有一个特定的逻辑次序。例如,可以以不同的次序来搜索 DSA。

注 2:此处规定的差错优先级规则仅适用于目录作为一个整体提供的抽象服务。当考虑目录的内部结构时,将应用

不同的规则。

如果目录同时检测到多个差错，那么以下清单将确定报告哪个差错。清单中级别较高的差错比级别较低的差错具有更高的逻辑优先级，报告级别较高的差错。

a) nameError；
b) updateError；
c) attributeError；
d) securityError；
e) serviceError。

以下差错不会引起任何优先级冲突：

a) abandonFailed，原因是它只针对放弃这一操作，不会遇到任何其他差错；
b) abandoned，如果与差错检测同时接收到放弃操作，那么不报告它。在这种情况下，返回带问题tooLate 的abandonFailed，并报告遇到的实际差错；
c) referral，它不是一个“真实的”差错，只是指出目录已经检测到 DUA 应将其请求提交给另一个访问点。

12.2 放弃

如果 DUA 利用适当的 InvokeId 调用一个放弃操作，那么对任何未完成的目录查询操作（即读、搜索、比较、列表）都可以报告该结果。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 中 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
abandoned ERROR ::={--不是字面上的“错误”
        PARAMETER     OPTIONALLY-PROTECTED{
            SET{COMPONENTS OF    CommonResults}}
        CODE      id-errocde-abandoned}
```

如果目录对差错进行标记、加密或者标记和加密，那么SecurityParameters 组件（见 7.10）将包括在 CommonResults 中（见 7.4）。

12.3 放弃失败

abandonFailed 差错用于报告在尝试放弃一个操作过程中遇到的问题。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 中 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
abandonFailed ERROR ::={
        PARAMETER                    OPTIONALLY-PROTECTED{
            SET {
                problem              [0]     AbandonProblem,
                operation            [1]     InvokeId,
                COMPONENTS OF        CommonResults}}
        CODE                 id-errcode-abandonFailed}
AbandonProblem ::=INTEGER{noSuchOperation (1),tooLate(2),cannotAbandon(3)}
```

各个参数具有以下含义。

规定遇到的特殊problem。可指出以下任何问题：

a) noSuchOperation：当目录不了解将被放弃的操作时（可由于未发生此类调用，或由于目录忘了它）；
b) tooLate：当目录已对操作做出响应时；
c) cannotAbandon：当已尝试放弃一个操作时（对之是禁止的，如修改），或者当无法执行放弃时。

将放弃标识特定的operation（调用）。

如果目录对差错进行标记、加密或者标记和加密，SecurityParameters 组件（见 7.10）将包括在 CommonResults 中（见 7.4）。

通过使用CommonResults 的notification 组件，可以任选地对差错问题提供的信息做出限制。

12.4 属性差错

attributeError 用于报告一个与属性有关的问题。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 的 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
attributeError ERROR ::={
        PARAMETER       OPTIONALLY-PROTECTED {
            SET{
                object      [0]  Name,
                problems    [1]  SET OF SEQUENCE{
                    problem     [0]  AttributeProblem,
                    type        [1]  AttributeType,
                    value       [2]  AttributeValue OPTIONAL },
                COMPONENTS OF CommonResults } }
        CODE        id-errcode-attributeError }

AttributeProblem ::=INTEGER {
        noSuchAttributeOrValue              (1),
        invalidAttributeSyntax              (2),
        undefinedAttributeType              (3),
        inappropriateMatching               (4),
        constraintViolation                 (5),
        attributeOrValueAlreadyExists       (6),
        contextViolation                    (7) }
```

各个参数具有以下含义。

object 参数用于确定当发生差错时操作正在使用哪个条目。返回的名（称）可以只包括有关包含多个不同值（由上下文区分）的属性的主要不同值（即 DSA 不必使用上下文选择，如 7.7 所述，如同它对成功的操作所做的那样）。

可以规定一个或多个problems。每个problem（在下面确定）伴随一个属性type 指示，如果需要避免模糊性，那么value 将引起以下问题：

a) noSuchAttributeOrValue：命名的条目缺少一个属性或属性值，它规定为操作的一个变元；

b) invalidAttributeSyntax：规定为操作的一个变元的假设属性值不符合属性类型的属性语法；

c) undefinedAttributeType：提供了一个未定义的属性类型，作为操作的一个变元。该差错只可能发生在与addEntry 或modifyEntry 相关的操作中；

d) inappropriateMatching：尝试使用一个未为相关属性类型定义的匹配规则，如在一个过滤器中；

e) constraintViolation：在操作变元中提供的属性值不符合 GB/T 16264.2—2008 或属性定义所提的约束要求（例如，值超过了允许的最大值）；

f) attributeOrValueAlreadyExists：尝试增加一个已经在条目中存在的属性，或者一个已经在属性中存在的值；

g) contextViolation：在操作变元中随属性值提供的上下文清单或上下文不符合 GB/T 16264.2—2008 或上下文定义（例如，上下文值不符合正确的语法）或 DIT 上下文使用所提的约束要求。

如果目录对差错进行标记、加密或者标记和加密，SecurityParameters 组件（见 7.10）将包括在 CommonResults 中（见 7.4）。

通过使用CommonResults 的notification 组件，可以任选地对差错问题提供的信息做出限制。

12.5 名（称）差错

nameError 用于报告一个与名（称）有关的问题，提供的名（称）作为操作的变元。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 的 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
nameError ERROR ::={
    PARAMETER    OPTIONALLY-PROTECTED {
        SET {
            problem                 [0]  NameProblem,
            matched                 [1]  Name,
            COMPONENTS OF           CommonResults } }
    CODE                id-errcode-nameError }
NameProblem ::=INTEGER {
    noSuchObject                (1),
    aliasProblem                (2),
    invalidAttributeSyntax      (3),
    aliasDereferencingProblem   (4),
    contextProblem              (5)}
```

各个参数具有以下含义。

遇到特殊的problem。可能指出任何以下问题：

a) noSuchObject：提供的名（称）不匹配任何客体的名（称）；

b) aliasProblem：别名已解除引用，它未命名任何客体；

c) invalidAttributeSyntax：名（称）中 AVA 中的属性类型及其伴随的属性值不兼容；

d) aliasDereferencingProblem：在不允许别名的地方或者拒绝访问的地方遇到了别名；

e) contextProblem：名（称）中所用的上下文类型或值无法理解或无效、上下文变元名（称）的使用不可接受，或者在名（称）解析期间，一个声称名匹配于多个 DIT 条目的名（称）。

matched 参数包含匹配的 DIT 中最低条目（客体或别名）的名（称），并且是所提供名（称）的简短形式，或者如果已经解除引用别名，那么是结果名（称）的简短形式。返回的名（称）可以只包括有关包含多个不同值（由上下文区分）的属性的主要不同值（即 DSA 不必使用上下文选择，如 7.7 所述，如同它对成功的操作所做的那样）。

注：如果在目录操作变元提供的名（称）中存在属性类型与/或值问题，那么它通过带问题invalidAttributeSyntax 的 nameError 来报告，而不是作为attributeError 或updateError。

如果目录对差错进行标记、加密或者标记和加密，SecurityParameters 组件（见 7.10）将包括在 CommonResults 中（见 7.4）。

通过使用CommonResults 的notification 组件，可以任选地对差错问题提供的信息做出限制。

12.6 转向推荐

referral 操作用于将服务用户重新引导至一个或多个更适于完成所请求操作的访问点上。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 的 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
referral ERROR ::={不是文字上的“差错”
        PARAMETER OPTIONALLY-PROTECTED{
```

```
        SET{
            candidate          [0]    ContinuationReference,
            COMPONENTS OF      CommonResults}}
    CODE                id-errcode-referral}
```

差错有单个参数，它包含一个ContinuationReference，用于推动操作（见 GB/T 16264.4—2008）。

如果 DSA 对应一个 LDAP 请求，那么ContinuationReference 中的accessPoints 组件将包含一个有效的LabeledURI 值，在这种情况下，它将使用该值来创建一个 LDAP 转向推荐。如果它不包含一个有效的LabeledURI 值，那么它将不返回一个转向推荐。

如果目录对差错进行标记，那么SecurityParameters 组件（见 7.10）将包括在CommonResults 中（见 7.4）。

在进行连续引用之前，DUA 将检查与连续引用产生的请求完全相同的请求还未作为处理相同用户请求的一部分发布。如果已经发布，那么 DUA 将不进行连续引用，以避免循环引用。

12.7 安全差错

securityError 用于报告在因安全原因而执行的操作中出现的问题。如果操作参数由请求方进行标记、加密或者标记和加密（见 GB/T 16264.2—2008 的 17.3），那么目录可以对差错参数进行标记、加密或者标记和加密。

```
securityError ERROR ::={
    PARAMETER       OPTIONALLY-PROTECTED{
        SET {
            problem             [0]  SecurityProblem,
            spkmInfo            [1]  SPKM-ERROR,
            COMPONENTS OF  CommonResults } }
    CODE            id-errcode-securityError }
SecurityProblem ::=INTEGER {
    inappropriateAuthentication     (1),
    invalidCredentials              (2),
    insufficientAccessRights        (3),
    invalidSignature                (4),
    protectionRequired              (5),
    noInformation                   (6),
    blockedCredentials              (7),
    invalidQOPMatch                 (8),
    spkmError                       (9) }
```

差错有一个单个参数，用于报告遇到的特殊problem。可以指出以下问题：

a) inappropriateAuthentication：与请求方证书关联的安全级别与请求的保护级别不一致，例如，提供的是简单的证书，而要求的是增强的证书；

b) invalidCredentials：提供的证书无效；

c) insufficientAccessRights：请求方无权完成请求的操作；

d) invalidSignature：发现请求签名无效；

e) protectionRequired：目录不愿完成请求的操作，原因是变元未标记；

f) noInformation：请求的操作产生一个安全差错，对它没有任何信息可用；

g) blockedCredentials：证书因安全方面的因素而受阻（例如，因连续多次提供无效的口令）；返回该差错的决定由对 DSA 起作用的安全策略进行管理；

h) invalidQOPMatch:两个实体具有不同的保护参数,分别为各自的安全服务而定义;

i) spkmError:发现提供的 SPKM 令牌无效。spkmInfo 参数包含一个指示,指出这是一个 SPKM 差错令牌,以及与该差错关联的 SPKM 上下文标识符。

如果目录对差错进行签名、加密或者签名和加密,SecurityParameters 组件(见 7.10)将包括在 CommonResults 中(见 7.4)

通过使用CommonResults 的notification 组件,可以任选地对差错问题提供的信息做出限制。

12.8 服务差错

serviceError 用于报告与服务提供有关的问题。如果操作参数由请求方进行签名、加密或者签名和加密(见 GB/T 16264.2—2008 的 17.3),那么目录可以对差错参数进行签名、加密或者签名和加密。

```
serviceError ERROR ::={
    PARAMETER       OPTIONALLY-PROTECTED {
        SET {
            problem             [0]     ServiceProblem,
            COMPONENTS OF               CommonResults } }
    CODE            id-errcode-serviceError }
ServiceProblem ::=INTEGER {
    busy                        (1),
    unavailable                 (2),
    unwillingToPerform          (3),
    chainingRequired            (4),
    unableToProceed             (5),
    invalidReference            (6),
    timeLimitExceeded           (7),
    administrativeLimitExceeded (8),
    loopDetected                (9),
    unavailableCriticalExtension (10),
    outOfScope                  (11),
    ditError                    (12),
    invalidQueryReference       (13),
    requestedServiceNotAvailable (14),
    unsupportedMatchingUse      (15),
    ambiguousKeyAttributes      ( 16),
    saslBindlnProgress          (17) }
```

差错有一个单个参数,用于报告遇到的特殊problem。可以指出以下问题:

a) busy:目录或其某部分目前太忙,无法执行请求的操作,但可以在短暂等待后执行。

b) unavailable:目录或其某部分目前无法使用。

c) unwillingToPerform:目录或其某部分目前尚未做好执行该请求的准备,例如,由于它将导致过量耗费资源或者与所涉及的管理权威部门的策略存在冲突。

d) chainingRequired:除了通过链接,目录无法完成请求,然而链接被chainingProhibited 服务控制选项所禁止。

e) unableToProceed:返回该差错的 DSA 对相应的命名上下文没有管理权限,结果是,无法参与名(称)解析。

f) invalidReference:无法执行 DUA 指向的请求,(通过OperationProgress)——这由使用无效的

转向推荐而引起。

g) timeLimitExceeded:目录到达了用户在服务控制中设置的时间限度。无任何结果可返回给用户。

h) administrativeLimitExceeded:目录到达了管理权威部门设置的某个限度,并且没有任何部分结果可返回给用户。

i) loopDetected:因内部循环,目录无法完成该请求。

j) unavailableCriticalExtension:因一个或多个临界扩展无法使用,目录无法完成请求。

k) outOfScope:在请求范围内没有任何转向推荐可用。

l) ditError:因 DIT 一致性问题,目录无法完成请求。

m) invalidQueryReference:请求操作的参数无效。如果分页结果中的queryReference 无效,那么报告该问题。

注:第 1 版本系统不支持该问题。

n) requestedServiceNotAvailable:由于没有任何搜索规则可用于搜索,或者由于搜索与所用的搜索规则发生冲突,因此服务特定管理区域内的搜索请求失败。可以与该服务问题一起返回额外的诊断信息。对不同情况下的此类额外信息在第 13 章中定义。

o) unsupportedMatchingUse:当设置了performExactly 搜索选项时,尝试使用 DSA 不支持的匹配规则,例如,在过滤器中。

p) ambiguousKeyAttributes:选择了基于映射的匹配规则,但可映射的过滤器项提供了多个违犯相关映射表的匹配。该差错情况伴随一个通告属性,由相关的、基于匹配的匹配规则指示。

q) saslBindInProgress:对某些鉴别机制,请求方可能需要多次调用directoryBind 操作。这通过响应方发送一个带问题saslBindInProgress 的serviceError 来指示。它指出,响应方要求请求方调用一个新的directoryBind 操作,利用同样的SaslCredentials 机制,继续进行鉴别过程。如果请求方希望能在任何阶段中断鉴别过程,那么它可以调用一个SaslAbort 设为TRUE 的directoryBind 操作。

如果目录对差错进行标记、加密或者标记和加密,SecurityParameters 组件(见 7.10)将包括在CommonResults 中(见 7.4)。

通过使用CommonResults 的notification 组件,可以任选地对差错问题提供的信息做出限制。

12.9 更新差错

updateError 用于报告与尝试增加、删除或修改 DIB 中信息有关的问题。如果操作参数由请求方进行标记、加密或者标记和加密(见 GB/T 16264.2—2008 的 17.3),那么目录可以对差错参数进行标记、加密或者标记和加密。

```
updateError ERROR ::= {
        PARAMETER       OPTIONALLY-PROTECTED {
                SET {
                        problem                 [0]   UpdateProblem,
                        attributeInfo           [1]   SET SIZE (1..MAX) OF CHOICE {
                                attributeType               AttributeType,
                                attribute                   Attribute } OPTIONAL,
                        COMPONENTS OF CommonResults } }
        CODE                    id-errcode-updateError }
UpdateProblem ::= INTEGER {
        namingViolation                 (1),
        objectClassViolation            (2),
```

notAllowedOnNonLeaf (3),
notAllowedOnRDN (4),
entryAlreadyExists (5),
affectsMultipleDSAs (6),
objectClassModificationProhibited (7),
noSuchSuperior (8),
notAncestor (9),
parentNotAncestor (10),
hierarchyRuleViolation (11),
familyRuleViolation (12) }

问题参数用于报告遇到的特殊问题。可以指出以下问题：

a) namingViolation：所做的增加或修改尝试将与目录方案和 GB/T 16264.2—2008 中定义的 DIT 结构规则发生冲突。也就是说，它将把一个条目作为别名条目的下级，或者在 DIT 区域内不允许其客体类成员，或者将为条目定义一个 RDN，以便包括禁止的属性类型；

b) objectClassViolation：所做的更新尝试将产生一个与条目内容规则不一致的条目，例如其客体类定义、DTI 内容规则，或者与客体类相关的、GB/T 16264.2—2008 中的各定义；

c) notAllowedOnNonLeaf：只允许对 DIT 的叶条目尝试操作；

d) notAllowedOnRDN：所做的操作尝试将对 RDN 产生影响（例如，移除作为 RDN 一部分的某个属性）；

e) entryAlreadyExists：尝试的addEntry 或modifyDN 操作，命名一个已经存在的条目；

注 1：这包括由 RDN 引起的冲突，它包括多个由上下文区分的不同值，而不管上下文是什么，如 GB/T 16264.2—2008 所述。

f) affectsMultipleDSAs：所做的更新尝试将需要在多个不允许该操作的 DSA 上进行；

g) objectClassModificationProhibited 一个尝试修改条目结构客体类的操作；

h) noSuchSuperior：尝试的modifyDN 操作，命名一个不存在的、新的上级条目；

i) notAncestor：尝试删除复合条目而不将祖（条目）规定为客体的操作；

j) parentNotAncestor：尝试将条目建成为非祖（条目）家族成员下直接层次下级的操作；

k) hierarchyRuleViolation：尝试打破适用于层次型组的规则的操作：一个层次型组必须完全处于任何服务特定管理区域外，或者必须完全包含在某个服务特定管理区域内；层次型组局限于单个 DSA；

l) familyRuleViolation：尝试打破适用于复合条目内各家族的规则的操作。

attributeInfo 参数用于确定引起问题的特殊属性类型和可能值。如果报告objectClassViolation，那么attribute 项将出现，指示引起问题的objectClass 属性类型，列出引起问题的客体类；还可以出现额外的attributeType 项（例如，用于确定丢失的强制性属性或外来属性）。

注 2：updateError 不是利用addEntry、removeEntry、modifyEntry 或modifyDN 操作中遇到的属性类型、值或约束冲突来报告问题。此类问题通过attributeError 来报告。

如果目录对差错进行标记、加密或者标记和加密，那么SecurityParameters 组件（见 7.10）将包括在 CommonResults 中（见 7.4）。

通过使用CommonResults 的notification 组件，可以任选地对差错问题提供的信息做出限制。

13 搜索变元的分析

本章只与在服务特定管理区域内开始其最初评估阶段的搜索操作有关。

本规程有两个目的：

a) 它提供了搜索确认功能(见 GB/T 16264.2—2008 中 16.12)。不过,搜索确认功能不产生差错信息。如果在规程运行期间遇到差错,那么评估停止并返回 FALSE,否则返回 TRUE。对空搜索规则的搜索确认将总返回 TRUE。

b) 当未找到任何管理-搜索-规则时,使用本规程,当标识单个搜索规则时,可以对SearchArgument 进行评估,以确定search 请求为什么失败。在这种情况下,当发现一个差错条件时,评估停止,在CommonResults 数据类型的notification 组件中提供必要的诊断信息,并返回一个带问题requestedServiceNotAvailable 的服务差错。包括什么样的诊断信息取决于确定的差错类型。

注:依据上述规定,可以利用相同的搜索规则对一个搜索请求做两次评估。如何优化它不在本规范的讨论范围内,而是一个执行决定。

本规程假设执行将不允许可调用的搜索规则:

——规定不支持的属性类型、上下文类型、匹配规则、匹配限制等;

——规定基于映射的匹配算法,对于搜索规则管理的搜索类型,它们不被支持或无关;

——规定与搜索规则冲突的匹配规则替换;

——指的是执行不支持的、可选的搜索规则特性;或者

——不一致的或差错的。

13.1 一般性检查搜索过滤器

评估首先检查过滤器是否与某些利用以下规程的基本限制有冲突:

a) 如果过滤器中描述的属性类型未由请求属性概貌在inputAttributeTypes 搜索规则组件中进行描述,那么notification 将包含:

——searchServiceProblem 通告属性,值为id-pr-searchAttributeViolation;

——serviceType 通告属性,值为搜索规则的serviceType 组件;以及

——attributeTypeList 通告属性,值为用于确定非法属性类型的客体标识符。

b) 如果存在只通过求反的过滤器项描述的属性类型,那么notification 将包含:

——searchServiceProblem 通告属性,值为id-pr-attributeNegationViolation;

——serviceType 通告属性,值为搜索规则的serviceType 组件;以及

——attributeTypeList 通告属性,值为用于确定在过滤器中被非法求反的属性类型的客体标识符。

c) 检查attributeCombination 中规定的条件是否满足有关属性类型非求反出现的要求。如果强制性属性类型,即需无条件地由非求反的过滤器项在过滤器中描述的属性类型,未出现在任何分过滤器中,那么notification 将包含:

——searchServiceProblem 通告属性,值为id-pr-missingSearchAttribute;

——serviceType 通告属性,值为搜索规则的serviceType 组件;以及

——attributeTypeList 通告属性,值为用于确定丢失属性类型的客体标识符。

如果要求的组合未出现,那么notification 将包含:

——searchServiceProblem 通告属性,值为id-pr-searchAttributeCombinationViolation;

——serviceType 通告属性,值为搜索规则的serviceType 组件;以及

——用于确定丢失组合的attributeCombinations 通告属性。

d) 对拥有selectedValues 子组件而值集为空的请求属性概貌,检查这些属性类型中是否存在不满足以下要求之一的过滤器项:

——过滤器项为present 类型,并且contexts 子组件未出现在请求属性概貌中;或者

——过滤器项为contextPresent 类型,并且contexts 子组件出现在请求属性概貌中。

如果上述检查对任何过滤器项都失败,那么notification 将包含:

——searchServiceProblem 通告属性，值为id-pr-searchValueNotAllowed；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——filterItem 通告属性，值为失败的过滤器项。

e) 对拥有contexts 子组件的请求属性概貌，检查上下文类型中是否存在不包括在该子组件中的过滤器项。如果存在，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchContextViolation；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——contextTypeList 通告属性，值为有关非法上下文类型的客体标识符。

f) 如果在搜索规则中采用了针对subset 组件的allowed 选项，那么检查SearchArgument 的subset 变元是否符合规定的要求。如果不复合，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchSubsetViolation；以及

——serviceType 通告属性，值为搜索规则的serviceType 组件。

13.2 检查请求属性概貌

如果上述规程未发生任何差错，那么需对每个分过滤器进行检查，看各分过滤器中所描述的任何属性类型是否也都有效地出现了。该规程不规定任何对分过滤器进行评估的次序。对有效出现在分过滤器中的属性类型，至少需要通过一个非求反的过滤器项对其进行描述，指出它符合相应请求属性概貌的要求。利用以下规程对非求反的过滤器项进行评估。

按以下次序对非求反的过滤器项进行检查：

a) 对每个分过滤器，对需无条件进行描述的属性类型的过滤器项进行检查；

b) 对每个分过滤器，对需条件进行描述的属性类型的过滤器项进行检查；以及

c) 对每个分过滤器，对剩余的过滤器项进行检查。

如果对分过滤器的评估失败，那么评估停止并返回差错信息，详述如下。

如果分过滤器中的属性类型通过若干非求反的过滤器项进行描述，那么原则上对每个这样的过滤器项都进行检查，直至找到一个符合要求的过滤器项，或对所有的过滤器项都进行了检查。如果在规程运行过程中有一个过滤器项失败，那么它留待进一步评估。由最后一个失败的属性类型过滤器项来决定返回的诊断信息。

利用以下程度对过滤器项进行评估：

a) 如果请求属性概貌中的selectedValues 组件不存在，或者它存在并非空，那么检查equality、substrings、approximateMatch 或extensibleMatch 类型的过滤器项。如果不这样，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchValueRequired；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——attributeTypeList 通告属性，值为用于从过滤器项中确定属性类型的客体标识符。

b) 如果相应请求属性概貌中的selectedValues 子组件存在并非空，那么检查过滤器项是否不匹配该子组件中规定的任何值。如果这样，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-invalidSearchValue；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——filterItem 通告属性，失败的过滤器项作为其唯一值。

c) 如果contexts 子组件未出现，那么继续下一个小条。

d) 检查contextCombination 子组件中规定的条件是否满足有关上下文类型的要求。如果强制性上下文类型丢失，即需无条件地为属性类型描述的上下文类型，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-missingSearchContext；

——serviceType 通告属性，值为搜索规则的serviceType 组件；

——attributeTypeList 通告属性，只有一个值，为用于从过滤器项中确定属性类型的客体标识符；以及

——contextTypeList 通告属性，值为用于确定丢失上下文类型的客体标识符。

如果要求的组合未出现，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchContextCombinationViolation；

——serviceType 通告属性，值为搜索规则的serviceType 组件；

——attributeTypeList 通告属性，只有一个值，为用于从过滤器项中确定属性类型的客体标识符；以及

——contextCombinations 通告属性，用于确定丢失的组合。

e) 检查分过滤器中的属性类型的上下文断言是否全部都包括在了contexts 子组件中。如果不这样，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchContextViolation；

——serviceType 通告属性，值为搜索规则的serviceType 组件；

——attributeTypeList 通告属性，只有一个值，为用于从过滤器项中确定属性类型的客体标识符；以及

——contextTypeList 通告属性，值为用于确定属性类型不允许的上下文类型的客体标识符。

f) 如果包括了请求属性概貌contexts 子组件中任何上下文类型的上下文值，那么检查在分过滤器中为属性类型规定的任何上下文断言是否都包含了未为contexts 子组件中相应上下文类型规定的值。如果这样，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-searchContextValueViolation；

——serviceType 通告属性，值为搜索规则的serviceType 组件；

——attributeTypeList 通告属性，只有一个值，为用于从过滤器项中确定属性类型的客体标识符；以及

——contextList 通告属性，值为属性类型不允许的上下文断言。

13.3 检查控制和层次选择

如果搜索请求未成功完成控制和层次选择的测试，如 GB/T 16264.2—2008 中所规定，那么将执行本条中的规程。

a) 如果搜索规则的defaultControls 组件或defaultControls 的hierarchyOptions 子组件不存在，并且搜索请求规定self 旁的层次选择，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-hierarchySelectForbidden；以及

——serviceType 通告属性，值为搜索规则的serviceType 组件。

b) 如果依据搜索规则中defaultControls 和mandatoryControls 组件的组合，不允许请求中的层次选择选项或丢失某些选择，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-invalidHierarchySelect；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——hierarchySelectList 通告属性，值为确定无效层次选择选项的比特束。

c) 如果 DSA 不支持请求中的层次选择选项，并且不被 b)涵盖，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-unavailableHierarchySelect；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——hierarchySelectList 通告属性，值为确定不支持层次选择选项的比特束。

d) 如果依据搜索规则中defaultControls 和mandatoryControls 组件的组合，不允许请求中的搜索控制选项(通过 10.2.1 定义)或丢失某些选项，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-invalidSearchControlOptions；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——searchControlOptionsList 通告属性，值为确定无效搜索控制选项的比特束。

e) 如果依据搜索规则中defaultControls 和mandatoryControls 组件的组合，不允许请求中的服务控制选项或丢失某些选项，那么notification 将包含：

——searchServiceProblem 通告属性，值为id-pr-invalidServiceControlOptions；

——serviceType 通告属性，值为搜索规则的serviceType 组件；以及

——serviceControlOptionsList 通告属性，值为确定无效服务控制选项的比特束。

13.4 检查匹配使用

在搜索确认规程中，本条代表确认中的最后一步，它假设search 请求已通过了所有其他确认步骤。最后一步失败的搜索规则置于MatchProblemSR（见 GB/T 16264.4—2008 的 19.3.2.2.1 的 c））清单中。

如果搜索请求不符合matchingUse 的要求，如 GB/T 16264.2—2008 的 16.10.2 中有关请求属性概貌的规定，那么有关其中之一失败的请求属性概貌的notification 将包含：

——如果匹配限制冲突，那么是一个带值id-pr-attributeMatchingViolation 的searchServiceProblem 通告属性，如果匹配规则以一种不支持的方式进行应用，那么是一个带值id-pr-unsupportedMatchingUse 的 searchServiceProblem 通告属性；

——serviceType 通告属性，值为搜索规则的serviceType 组件；

——attributeTypeList 通告属性，值只为确定属性类型的客体标识符；以及

——对冲突的匹配限制，应包含额外的通告属性，由规范为该匹配限制规定。

注：当有若干request-attribute-profiles 无法确认时，那么选择哪个来创建一个notification 将是一件本地的事情。

附 录 A
（规范性附录）
用 ASN.1 描述的抽象服务

本附录包括本目录规范中所含的所有 ASN.1 类型、值和信息客体定义，其形式为 ASN.1 模块。DirectoryAbstractService。

```
DirectoryAbstractService {joint-iso-itu-t ds(5) module(1) directoryAbstractService(2) 5}
DEFINITIONS ::=
BEGIN
--EXPORTS All--
——输出本模块中定义的类型和值用于目录规范中所含的其他 ASN.1 模块，并供其他将使用它们
——访问目录服务的应用使用。其他应用可以将它们用于其自身目的，但这不会限制维护或改进目录；
——服务所需的扩展和修改。
IMPORTS
——来自 GB/T 16264.2—2008
    attributeCertificateDefinitions, authenticationFramework, basicAccessControl,
    commonProtocolSpecification, directoryShadowAbstractService, distributedOperations,
    enhancedSecurity, id-at, informationFramework, selectedAttributeTypes, serviceAdministration,
    upperBounds
        FROM UsefulDefinitions {joint-iso-itu-t ds(5) module(1) usefulDefinitions(0) 5}
    Attribute, ATTRIBUTE, AttributeType, AttributeTypeAssertion, AttributeValue,
    AttributeValueAssertion, CONTEXT, ContextAssertion, DistinguishedName,
    MATCHING-RULE, Name, OBJECT-CLASS, RelativeDistinguishedName,
    SupportedAttributes, SupportedContexts
        FROM InformationFramework informationFramework

    RelaxationPolicy
        FROM ServiceAdministration serviceAdministration

    AttributeTypeAndValue
        FROM BasicAccessControl basicAccessControl

    OPTIONALLY-PROTECTED{ }, OPTIONALLY-PROTECTED-SEQ{ }
        FROM EnhancedSecurity enhancedSecurity

——来自 GB/T 16264.4—2008
    AccessPoint, ContinuationReference, Exclusions, OperationProgress, ReferenceType
        FROM DistributedOperations distributedOperations

——来自 GB/T 16264.5—2008
    Code, ERROR, id-errcode-abandoned, id-errcode-abandonFailed, id-errcode-attributeError,
```

```
    id-errcode-nameError,id-errcode-referral,id-errcode-securityError,id-errcode-serviceError,
    id-errcode-updateError,id-opcode-abandon,id-opcode-addEntry,id-opcode-compare,
    id-opcode-list,id-opcode-modifyDN,id-opcode-modifyEntry,id-opcode-read,
    id-opcode-removeEntry,id-opcode-search,InvokeId,OPERATION
        FROM CommonProtocolSpecification commonProtocolSpecification

---来自 GB/T 16264.6—2008
    DirectoryString {}
        FROM SelectedAttributeTypes selectedAttributeTypes

    ub-domainLocalID,ub-saslMechanism
        FROM UpperBounds upperBounds

---来自 ISO/IEC 9594-8:2005
    AlgorithmIdentifier,CertificationPath,ENCRYPTED {},SIGNATURE {},SIGNED {}
        FROM AuthenticationFramework authenticationFramework

    AttributeCertificationPath
        FROM AttributeCertificateDefinitions attributeCertificateDefinitions

---来自 ISO/IEC 9594-9:2005
    AgreementID
        FROM DirectoryShadowAbstractService directoryShadowAbstractService

---来自 RFC 2025
    SPKM-ERROR,SPKM-REP-TI,SPKM-REQ
        FROM SpkmGssTokens { iso (1)identified-organization (3) dod(6)internet (1)
            security (5) mechanisms (5) spkm (1) spkmGssTokens (10) } ;

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

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

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

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

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

ServiceControlOptions ::= BIT STRING {
                preferChaining          (0),
                chainingProhibited      (1),
                localScope              (2),
                dontUseCopy             (3),
                dontDereferenceAliases  (4),
                subentries              (5),
                copyShallDo             (6),
                partialNameResolution   (7),
                manageDSAIT             (8),
                noSubtypeMatch          (9),
                noSubtypeSelection      (10),
```

```
            countFamily                 (11),
            dontSelectFriends           (12),
            dontMatchFriends            (13) }

EntryInformationSelection ::= SET {
    attributes                          CHOICE {
            allUserAttributes       [0]  NULL,
            select                  [1]  SET OF AttributeType
            --空集意味着不需要任何属性--}DEFAULT allUserAttributes:NULL,
    infoTypes                       [2]    INTEGER {
            attributeTypesOnly                  (0),
            attributeTypesAndValues             (1) } DEFAULT attributeTypesAndValues,
    extraAttributes                     CHOICE {
             allOperationalAttributes   [3]  NULL,
             select                     [4]  SET SIZE (1..MAX) OF AttributeType } OPTIONAL,
    contextSelection                ContextSelection OPTIONAL,
    returnContexts                  BOOLEAN DEFAULT FALSE,
    familyReturn                    FamilyReturn DEFAULT
                                    { memberSelect contributingEntriesOnly } }

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

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

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

EntryInformation ::= SEQUENCE {
    name                        Name,
    fromEntry                   BOOLEAN DEFAULT TRUE,
    information                 SET SIZE (1..MAX)OF CHOICE {
            attributeType           AttributeType,
            attribute               Attribute } OPTIONAL,
```

```
        incompleteEntry         [3]  BOOLEAN DEFAULT FALSE,--不在第一版本系统中
        partialName             [4]  BOOLEAN DEFAULT FALSE,--不在第一或第二版本系统中
        derivedEntry            [5]  BOOLEAN DEFAULT FALSE--不在第四版本之前的系统中--}

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

FamilyEntries ::=SEQUENCE {
        family-class            OBJECT-CLASS.&id,--结构客体类别值
        familyEntries           SEQUENCE OF FamilyEntry )

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

Filter ::=CHOICE {
      item  [0]  FilterItem,
      and   [1]  SET OF Filter,
      or    [2]  SET OF Filter,
      not   [3]  Filter }

FilterItem ::=CHOICE {
        equality        [0]   AttributeValueAssertion,
        substrings      [1]   SEQUENCE {
                type              ATTRIBUTE.&id ({ SupportedAttributes }),
                strings           SEQUENCE OF CHOICE {
                        initial       [0]  ATTRIBUTE.&Type
                                           ({ SupportedAttributes }{@substrings.type} ),
                        any           [1]  ATTRIBUTE.&Type
                                           ({ SupportedAttributes }{@substrings.type} ),
                        final         [2]  ATTRIBUTE.&Type
                                           ({ SupportedAttributes }{@substrings.type}),
                        control           Attribute } },--用于规定以下项的解释
        greaterOrEqual  [2]  AttributeValueAssertion,
        lessOrEqual     [3]  AttributeValueAssertion,
        present         [4]  AttributeType,
        approximateMatch [5] AttributeValueAssertion,
        extensibleMatch [6]  MatchingRuleAssertion,
```

```
        contextPresent      [7]  AttributeTypeAssertion}

MatchingRuleAssertion ::=SEQUENCE {
        matchingRule        [1]  SET SIZE (1..MAX) OF MATCHING-RULE.&id,
        type                [2]  AttributeType OPTIONAL,
        matchValue          [3]  MATCHING-RULE.&AssertionType ( CONSTRAINED BY {
                                 --matchValue 应是一个类型值,由matchingRule 确定的其中一个
                                 --MATCHING-RULE 的 &AssertionType 字段规定。--}),
        dnAttributes        [4]  BOOLEAN DEFAULT FALSE }

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

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

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

ProtectionRequest ::=INTEGER {none(0),signed (1) }
Time ::=CHOICE {
        utcTime             UTCTime,
        generalizedTime     GeneralizedTime }

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

--绑定和解绑定操作--
```

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

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

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

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

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

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

SaslCredentials ::=SEQUENCE {
      mechanism           [0]  DirectoryString { ub-saslMechanism },
      credentials         [1]  OCTET STRING OPTIONAL,
```

```
        saslAbort               [2]  BOOLEAN DEFAULT FALSE}

Token ::=SIGNED { SEQUENCE {
        algorithm               [0]  AlgorithmIdentifier,
        name                    [1]  DistinguishedName,
        time                    [2]  UTCTime,
        random                  [3]  BIT STRING,
        response                [4]  BIT STRING OPTIONAL,
        bindIntAlgorithm        [5]  SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindIntKeyInfo          [6]  BindKeyInfo OPTIONAL,
        bindConfAlgorithm       [7]  SEQUENCE SIZE (1..MAX) OF AlgorithmIdentifier OPTIONAL,
        bindConfKeyInfo         [8]  BindKeyInfo OPTIONAL } }

Versions ::=BIT STRING {v1(0),v2(1) }

DirectoryBindResult ::=DirectoryBindArgument

directoryBindError ERROR ::={
      PARAMETER                 OPTIONALLY-PROTECTED {
            SET {
                  versions              [0]  Versions DEFAULT {v1},
                  error                      CHOICE {
                        serviceError              [1]  ServiceProblem,
                        aecurityError             [2]  SecurityProblem }}}}

BindKeyInfo ::=ENCRYPTED { BIT STRING }

--操作、变元和结果--
read OPERATION ::={
      ARGUMENT          ReadArgument
      RESULT            ReadResult
      ERRORS            { attributeError|nameError|serviceError|referral|abandoned|
                        securityError }
      CODE              id-opcode-read }

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

ReadResult ::=OPTIONALLY-PROTECTED {
```

```
    SET{
        entry                   [0]  EntryInformation,
        modifyRights            [1]  ModifyRights OPTIONAL,
        COMPONENTS OF           CommonResults } }

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

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

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

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

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

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

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

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

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

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

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

```
        lowEstimate          [8]  INTEGER,
        exact                [9]  INTEGER } OPTIONAL,
    streamedResult           [10] BOOLEAN DEFAULT FALSE }

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

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

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

HierarchySelections ::= BIT STRING {
    self          (0),
    children      (1),
    parent        (2),
    hierarchy     (3),
    top           (4),
    subtree       (5),
    siblings      (6),
```

```
        siblingChildren   (7),
        siblingSubtree    (8),
        all               (9) }

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

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

DomainLocalID       ::= DirectoryString { ub-domainLocalID }

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

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

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

```
                COMPONENTS OF                CommonResults },
        uncorrelatedSearchInfo    [0]    SET OF SearchResult } }

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

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

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

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

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

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

modifyEntry OPERATION ::={
        ARGUMENT          ModifyEntryArgument
        RESULT            ModifyEntryResult
        ERRORS            { attributeError|nameError|serviceError|referral|securityError|
                          updateError }
```

```
        CODE                id-opcode-modifyEntry }

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

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

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

modifyDN OPERATION ::={
        ARGUMENT            ModifyDNArgument
        RESULT              ModifyDNResult
        ERRORS              { nameError|serviceError|referral|securityError|updateError }
        CODE                id-opcode-modifyDN }

ModifyDNArgument ::=OPTIONALLY-PROTECTED {
        SET {
            object          [0]     DistinguishedName,
            newRDN          [1]     RelativeDistinguishedName,
            deleteOldRDN    [2]     BOOLEAN DEFAULT FALSE,
            newSuperior     [3]     DistinguishedName OPTIONAL,
            COMPONENTS OF           CommonArguments } }

ModifyDNResult ::=CHOICE{
        null                NUL,
        information OPTIONALLY-PROTECTED-SEQ {
            SEQUENCE {
```

```
                newRDN                  ReiativeDistinguishedName,
                COMPONENTS OF           CommonResultsSeq } } }

--差错和参数--
abandoned ERROR ::={--不是字面上的"错误"
    PARAMETER    OPTIONALLY-PROTECTED {
            SET {COMPONENTS OF CommonResults} }
    CODE             id-errcode-abandoned }

abandonFailed ERROR ::={
    PARAMETER                   OPTIONALLY-PROTECTED {
        SET {
            problem             [0]     AbandonProblem,
            operation           [1]     Invokeld,
            COMPONENTS OF                CommonResults } }
        CODE          id-errcode-abandonFailed }

AbandonProblem ::=INTEGER { noSuchOperation (1),tooLate (2),cannotAbandon (3) }

attributeError ERROR::={
    PARAMETER    OPTIONALLY-PROTECTED {
        SET{
            object          [0]         Name,
            problems        [1]         SET OF SEQUENCE {
                problem                 [0]     AttributeProblem,
                type                    [1]     AttributeType,
                value                   [2]     AttributeValue OPTIONAL },
            COMPONENTS OF CommonResults } }
    CODE                id-errcode-attributeError }

AttributeProblem ::=INTEGER {
        noSuchAttributeOrValue          (1),
        invalidAttributeSyntax          (2),
        undefinedAttributeType          (3),
        inappropriateMatching           (4),
        constraintViolation             (5),
        attributeOrValueAlreadyExists   (6),
        contextViolation                (7) }

nameError ERROR ::={
    PARAMETER    OPTIONALLY-PROTECTED {
```

```
            SET {
                problem             [0]     NameProblem,
                matched             [1]     Name,
                COMPONENTS OF               CommonResults } }
        CODE                id-errcode-nameError }

NameProblem ::=INTEGER {
        noSuchObject                    (1),
        aliasProblem                    (2),
        invalidAttributeSyntax          (3),
        aliasDereferencingProblem       (4),
        contextProblem                  (5) }

referral ERROR ::={--不是文字上的"差错"
        PARAMETER OPTIONALLY-PROTECTED {
            SET {
                candidate           [0]     ContinuationReference,
                COMPONENTS OF               CommonResults} }
        CODE                id-errcode-referral }

securityError ERROR ::={
        PARAMETER       OPTIONALLY-PROTECTED {
            SET{
                problem             [0]     SecurityProblem,
                spkmlnfo            [1]     SPKM-ERROR,
                COMPONENTS OF               CommonResults } }
        CODE                id-errcode-securityError }

SecurityProblem ::=INTEGER {
        inappropriateAuthentication     (1),
        invalidCredentials              (2),
        insufficientAccessRights        (3),
        invalidSignature                (4),
        protectionRequired              (5),
        noInformation                   (6),
        blockedCredentials              (7),
        invalidQOPMatch                 (8),
        spkmError                       (9) }

serviceError ERROR ::={
        PARAMETER       OPTIONALLY-PROTECTED {
```

```
            SET {
                problem              [0]    ServiceProblem,
                COMPONENTS OF               CommonResults } }
        CODE            id-errcode-serviceError }

ServiceProblem ::=INTEGER {
        busy                                (1),
        unavailable                         (2),
        unwillingToPerform                  (3),
        chainingRequired                    (4),
        unableToProceed                     (5),
        invalidReference                    (6),
        timeLimitExceeded                   (7),
        administrativeLimitExceeded         (8),
        loopDetected                        (9),
        unavailableCriticalExtension        (10),
        outOfScope                          (11),
        ditError                            (12),
        invalidQueryReference               (13),
        requestedServiceNotAvailable        (14),
        unsupportedMatchingUse              (15),
        ambiguousKeyAttributes              (16),
        saslBindInProgress                  (17)}

updateError ERROR ::={
        PARAMETER       OPTIONALLY-PROTECTED{
                SET{
                    problem                 [0]     UpdateProblem,
                    attributeInfo           [1]     SET SIZE(1..MAX) OF CHOICE{
                        attributeType                   AttributeType,
                        attribute                       Attribute}OPTIONAL,
                    COMPONENTS OF           CommonResults}}
        CODE                    id-errcode-updateError}

UpdateProblem ::=INTEGER{
        namingViolation                     (1),
        objectClassViolation                (2),
        notAllowedOnNonLeaf                 (3),
        notAllowedOnRDN                     (4),
        entryAlreadyExists                  (5),
        affectsMultipleDSAs                 (6),
```

```
        objectClassModificationProhibited       (7),
        noSuchSuperior                          (8),
        notAncestor                             (9),
        parentNotAncestor                       (10),
        hierarchyRuleViolation                  (11),
        familyRuleViolation                     (12)}

--属性类型--
id-at-family-information        OBJECTIDENTIFIER      ::=      {id-at 64}
END--目录抽象服务
```

附 录 B
（资料性附录）
基本访问控制的操作语义

本附录包含许多个图表，用于描述与基本访问控制关联的语义，适用于目录操作的处理（见图 B.1～图 B.16）。

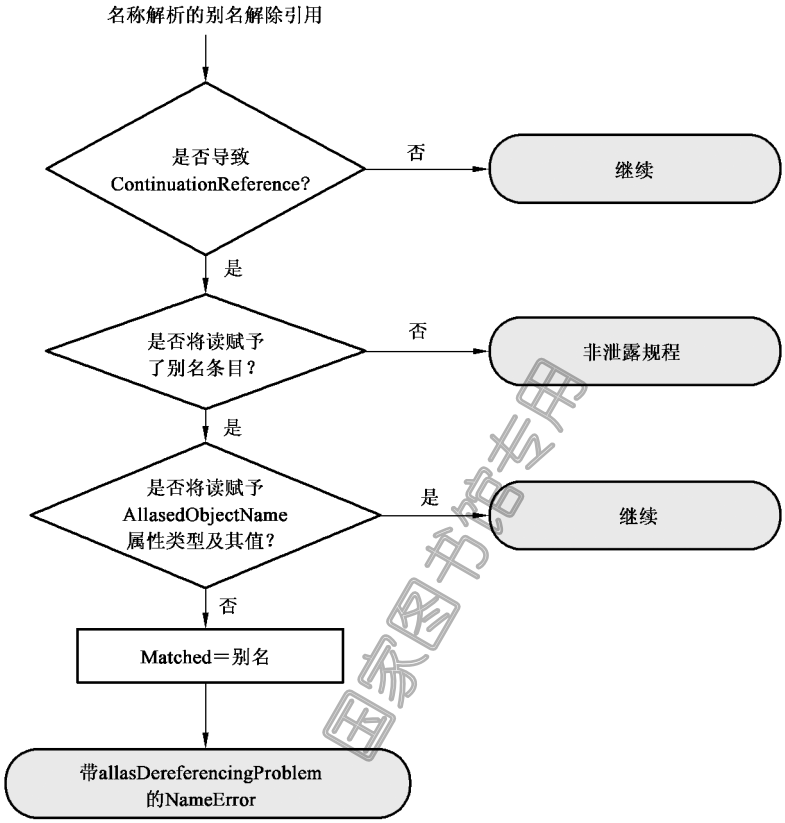

图 B.1 名（称）解析中别名解除引用

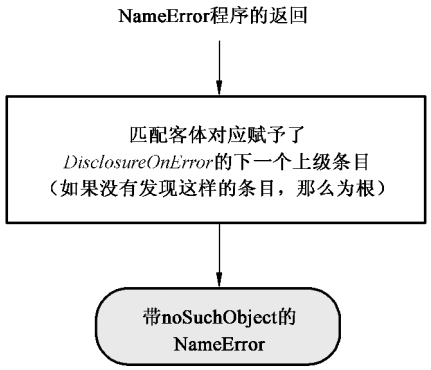

图 B.2 NameError 的返回

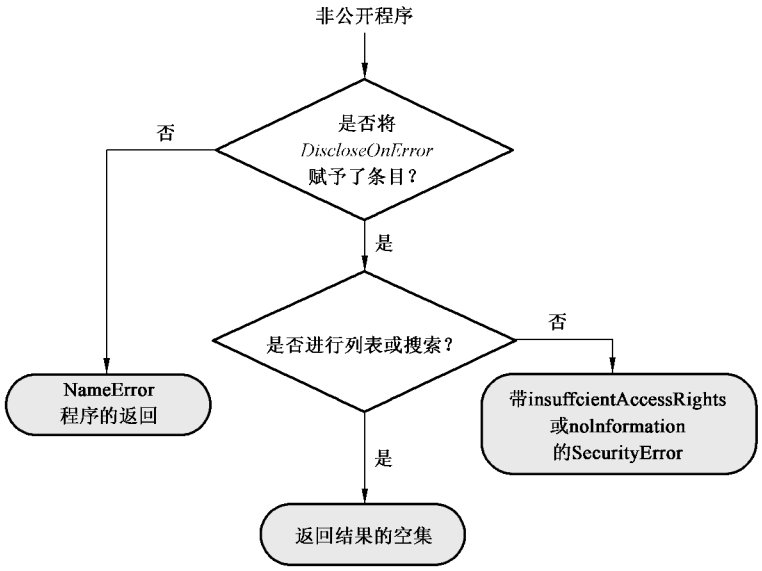

图 B.3　不公开条目是否存在

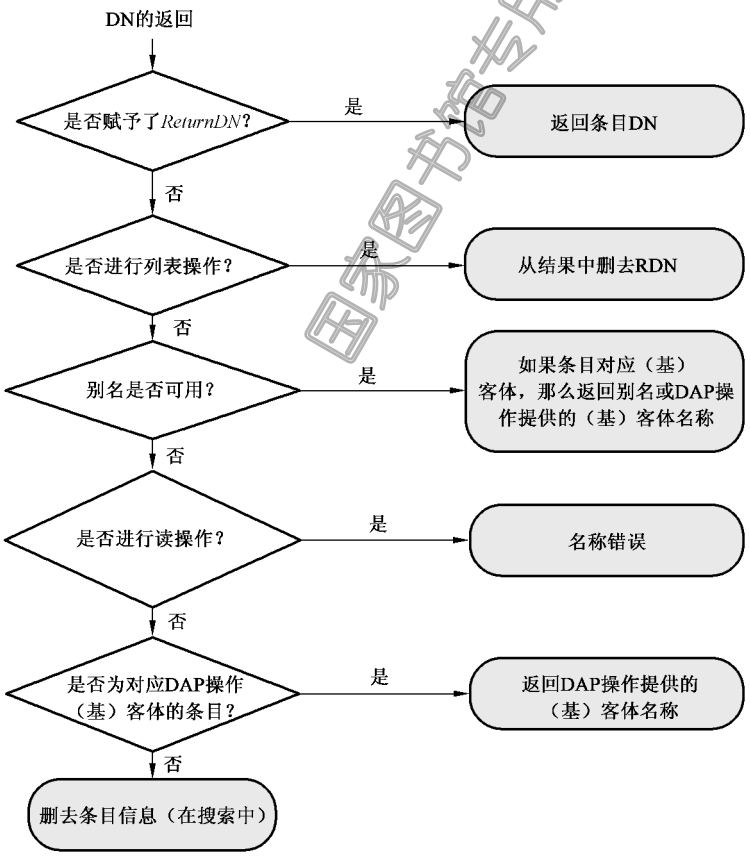

图 B.4　可辨别名的返回

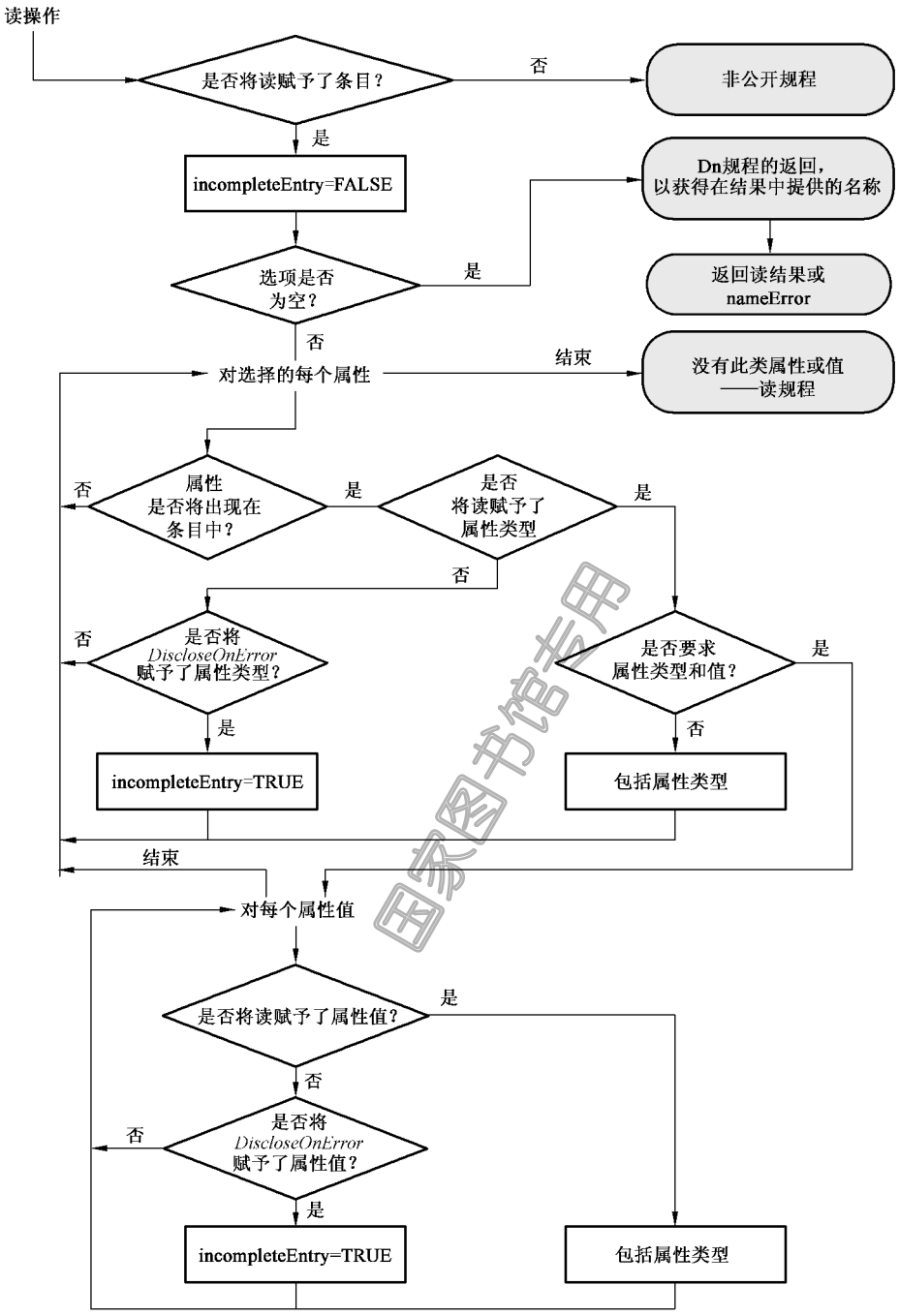

图 B.5　读操作

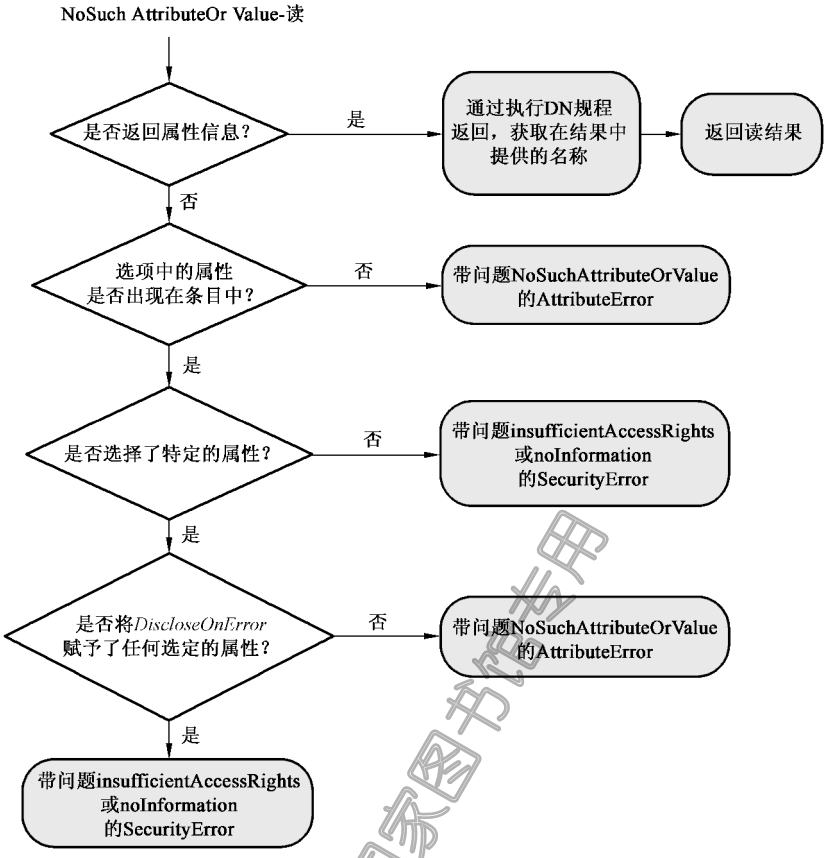

图 B.6　没有读的此类属性或值

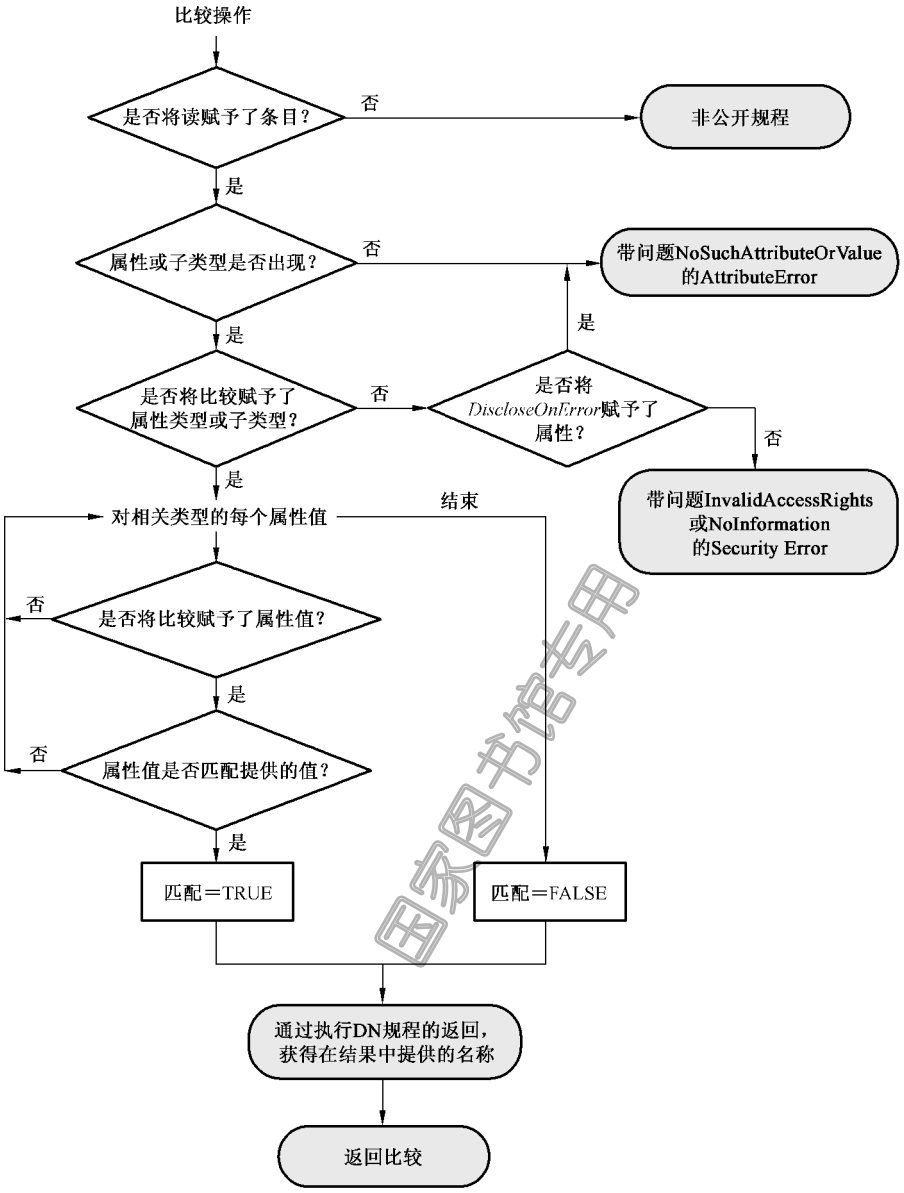

图 B.7　比较操作

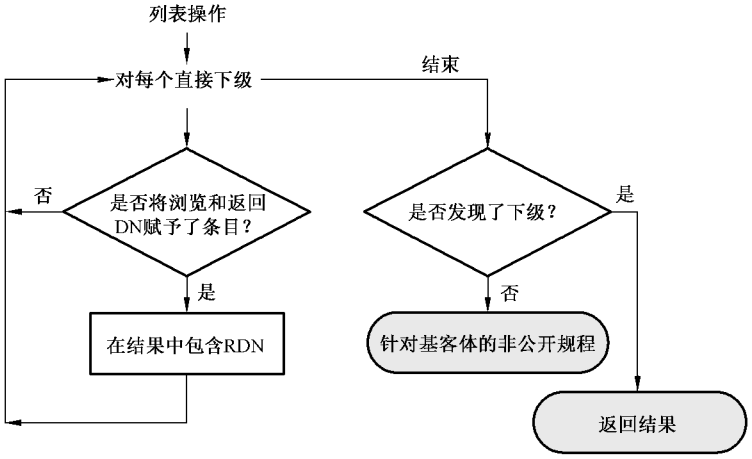

图 B.8 列表操作

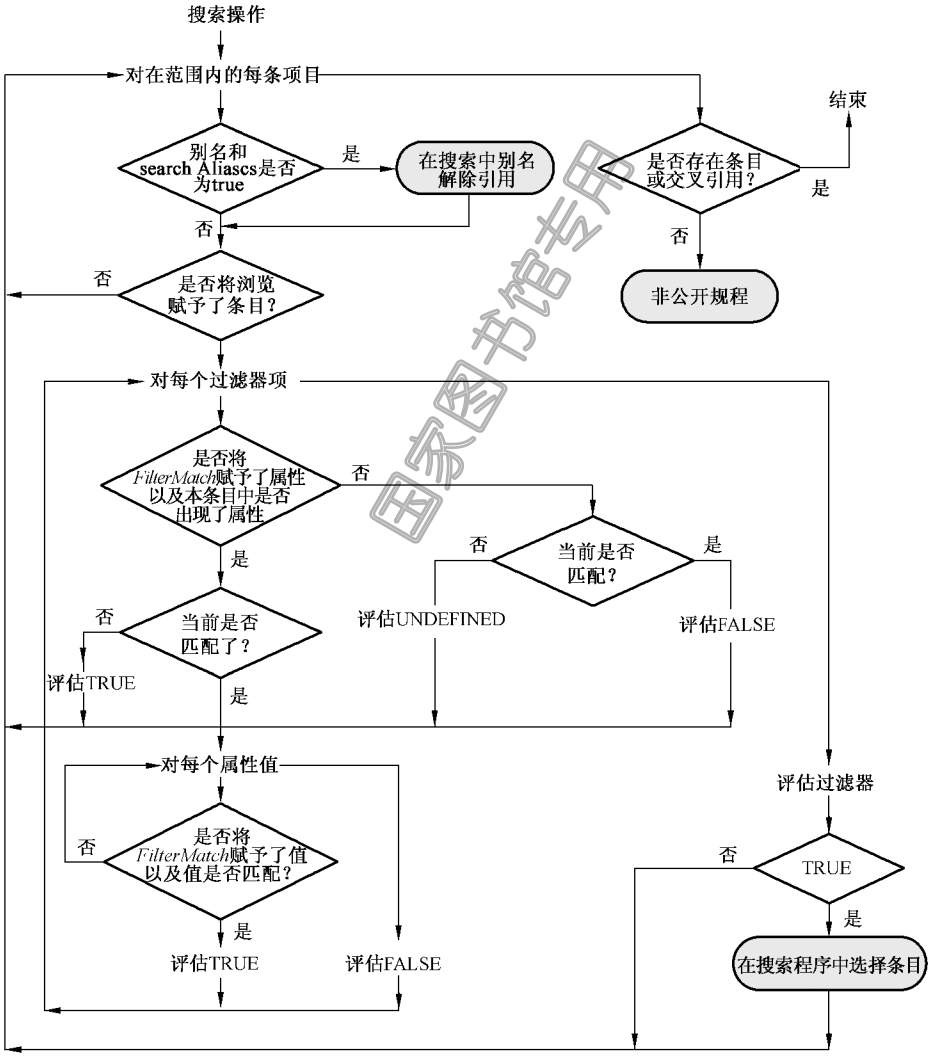

图 B.9 搜索操作

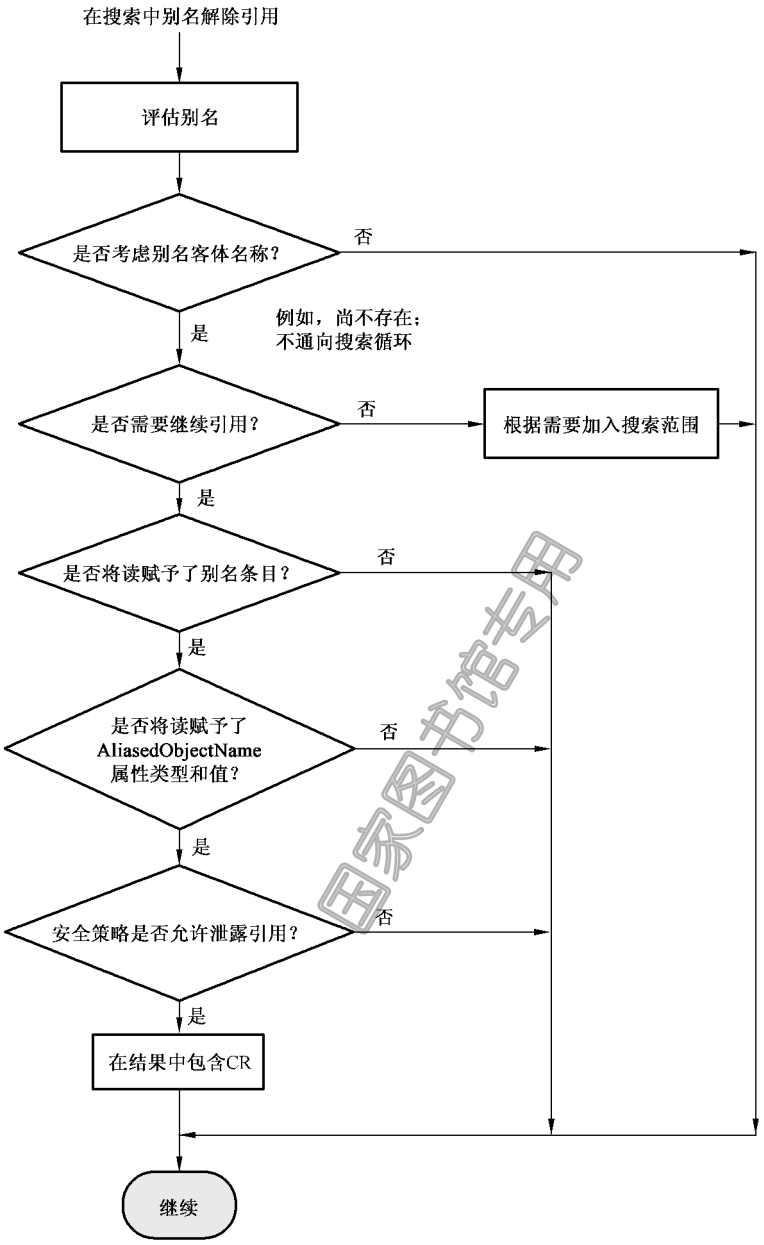

图 B.10　在搜索中别名解除引用

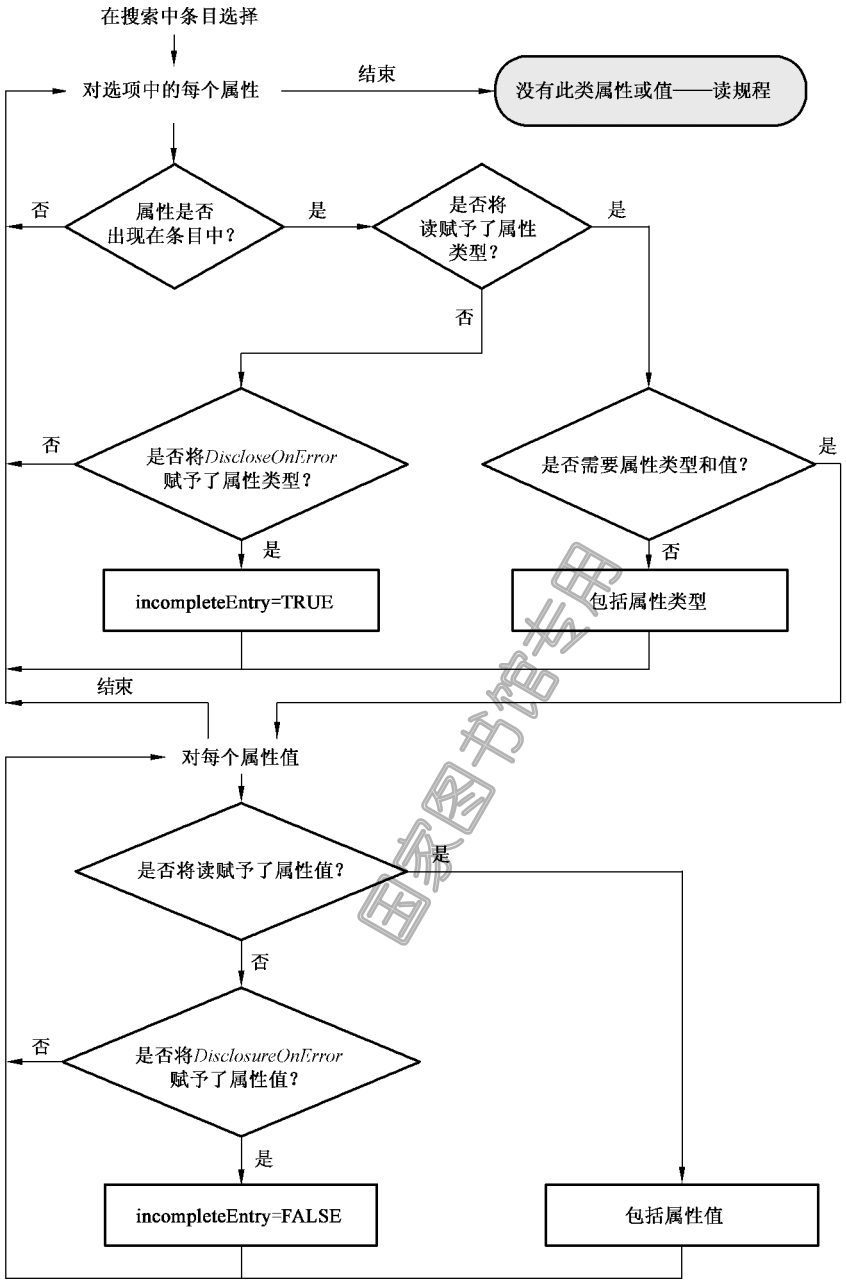

图 B.11　在搜索中条目选择

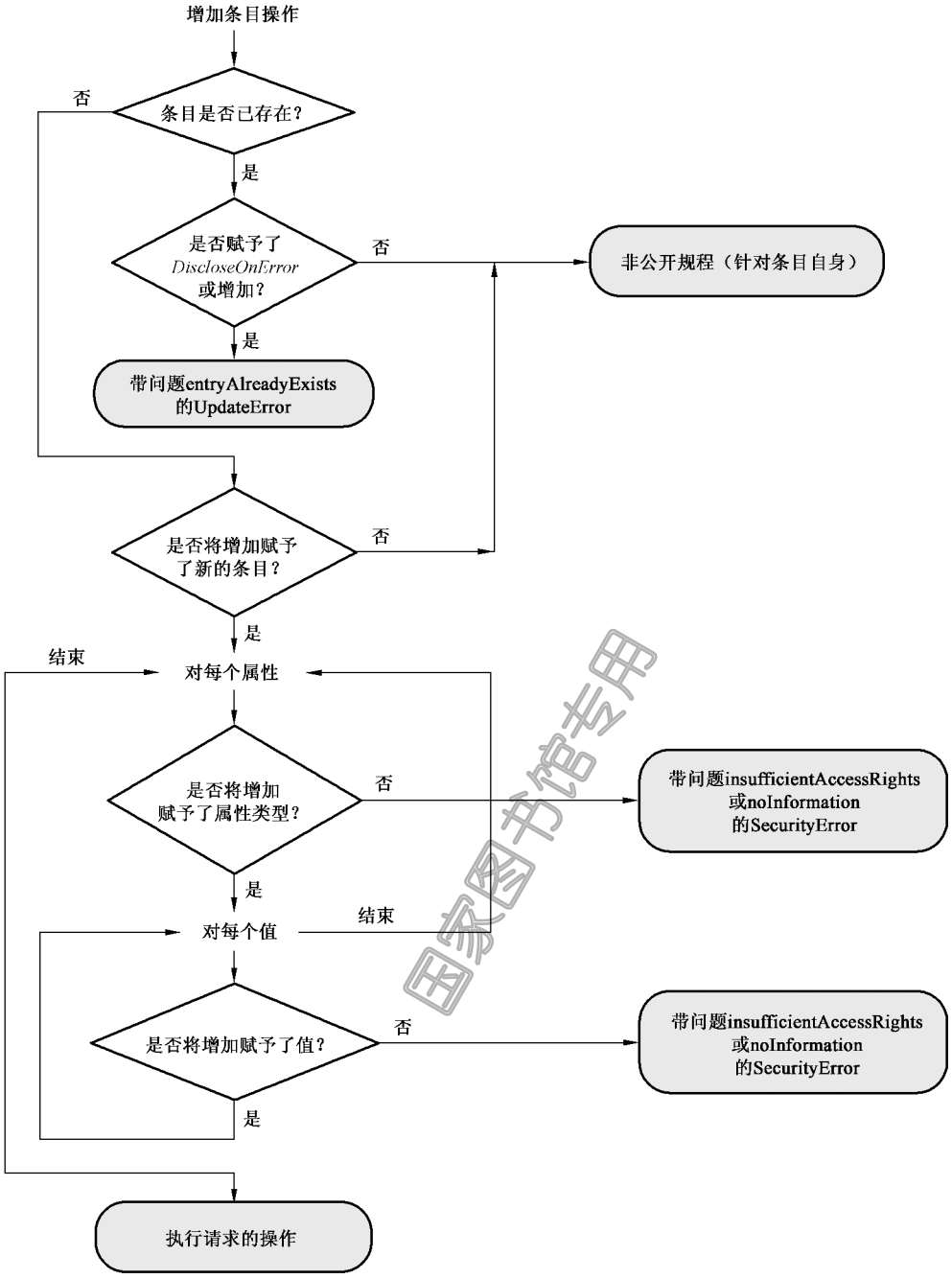

图 B.12　增加条目操作

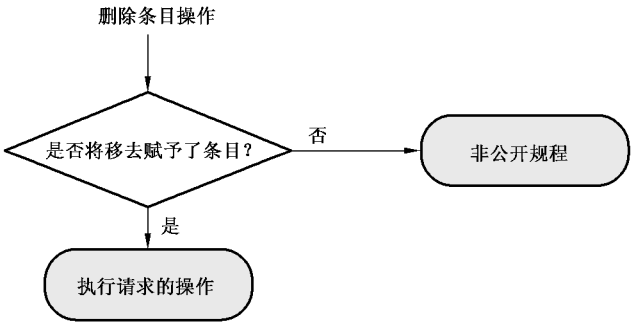

图 B.13　移除条目操作

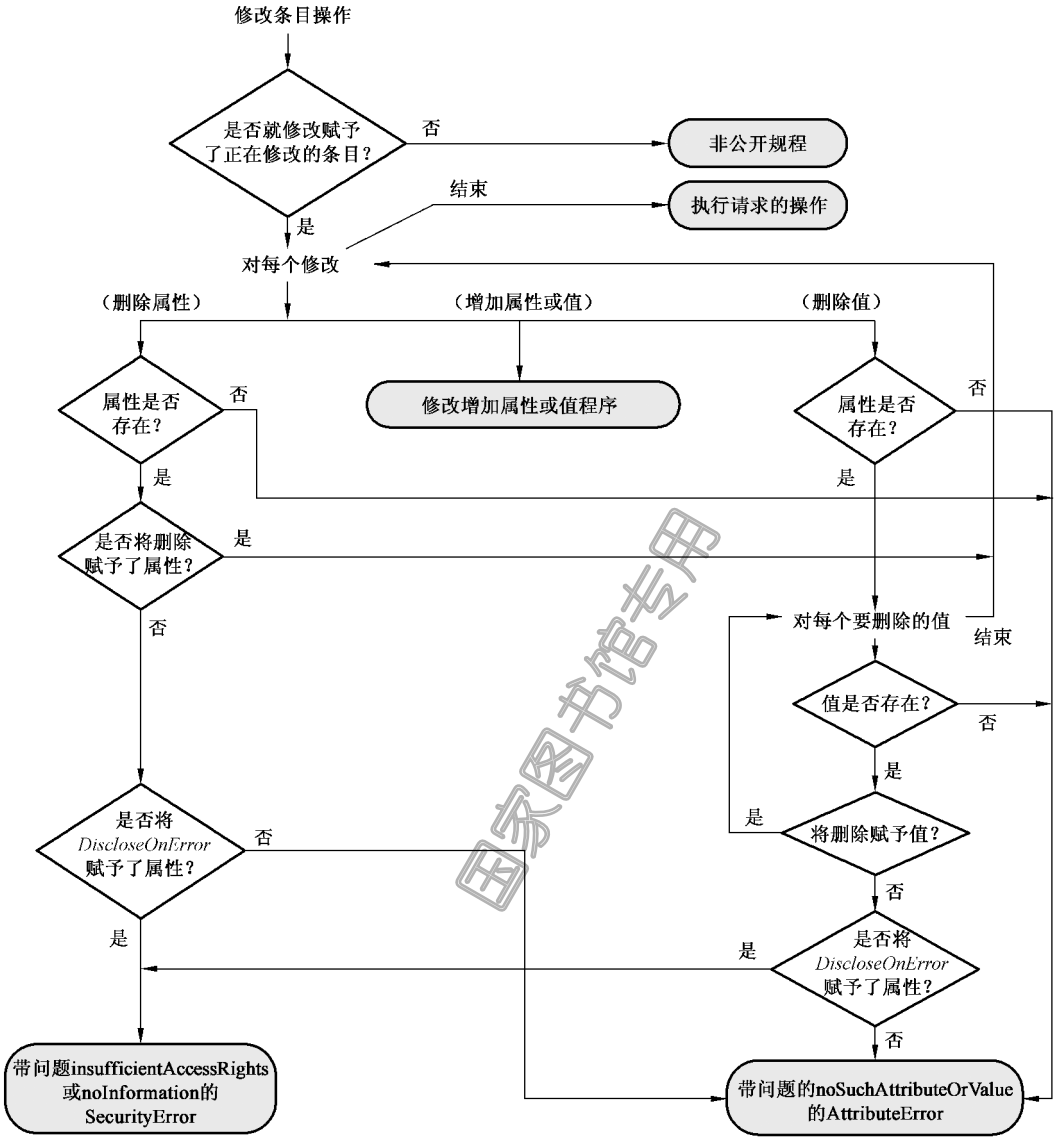

图 B.14　修改条目操作

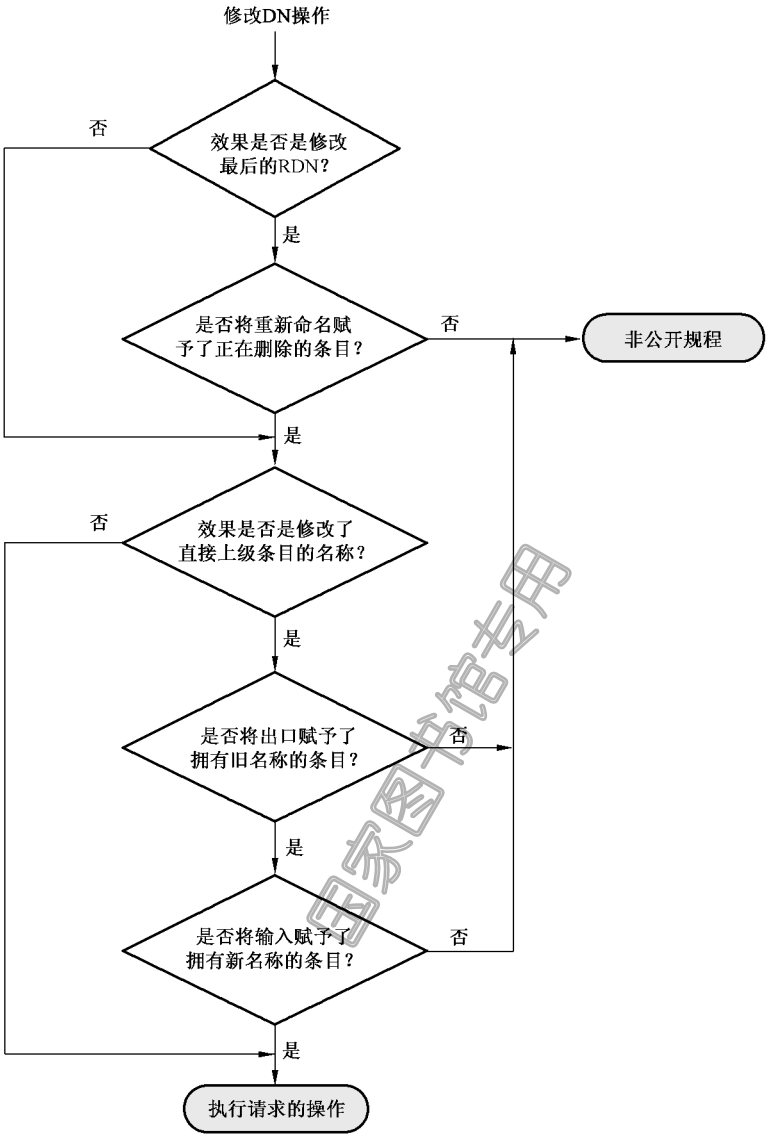

图 B.15 修改 DN 操作

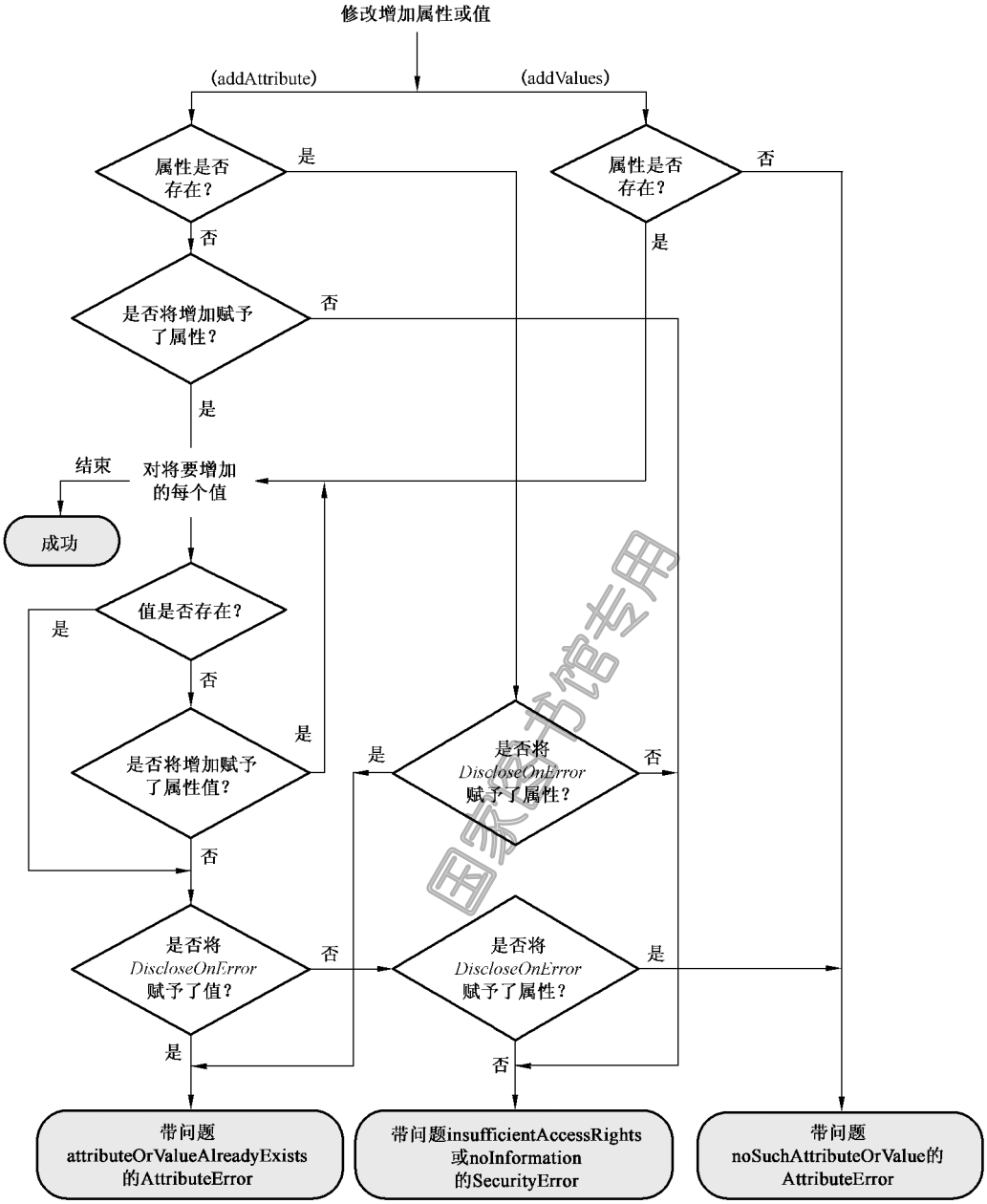

图 B.16　修改增加属性或值

附 录 C
（资料性附录）
搜索条目家族举例

C.1 单个家族举例

假设 Charles Smith 有多种通信模式：陆地电话、传真、移动电话和电子邮件，每种模式有其自身的关联参数。进一步假设 Charles Smith 有两个电子邮件账户，一个在其工作场所，一个在其家中，两个都提供了 POP3 邮箱和 SMTP 服务器。所有这些信息都可以存在于一个复合条目中，Charles Smith 的成员作为祖（条目），每种通信模式作为下级成员，每种电子邮件服务作为电子邮件通信模式的下级。这如下面的图 C.1 所示。由于作为祖（条目）直接下级的所有成员都有相同的结构客体类（comAddr），因此复合条目由一个单个的家族组成。

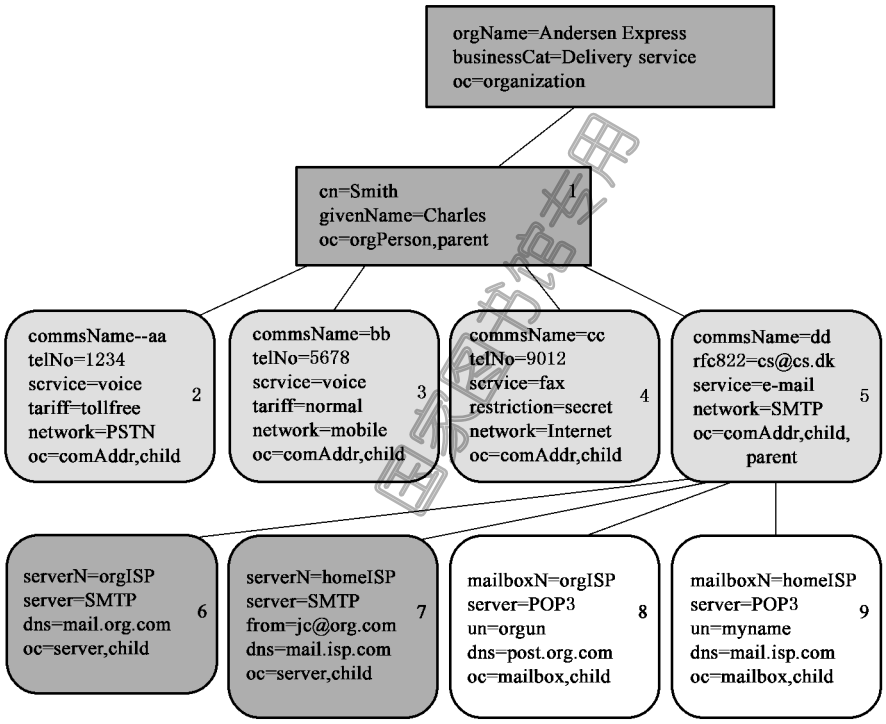

图 C.1 Charles Smith 的条目家族

假设search 请求由{…o=Andersen Express}的基本客体、{telNo=1234 & tariff=normal}的过滤器、wholeSubtree 或oneLevel 的子集产生。当familyGrouping 参数设为以下值时：

a) entryOnly：家族中将没有任何成员匹配于过滤器；

b) strands 或multiStrand：家族中将没有任何束或多束匹配于过滤器；

c) compoundEntry：成员 2 和成员 3 将一起匹配于过滤器，并将被标记为贡献成员。所有的成员都将被标记为参与成员。

对上面情况 a)和情况 b)，将不返回任何本复合条目中的内容。

对上面情况 c)，返回的信息将依赖于家族返回规范（如由EntryInformationSelection 中的familyReturn 给出）：

a) contributingEntriesOnly：各成员标记为贡献成员，即成员 2 和成员 3 将被返回；

b) participatingEntriesOnly 和compoundEntry：复合条目中的所有成员都将被返回。

C.2 多个家族举例

假设 Charles Smith 只有陆地电话和电子邮件，但还有两个具有关联参数的邮政地址。所有这些信息都可以存在于一个复合条目中，Charles Smith 的成员作为祖（条目），每种通信模式或每个邮政地址作为下级成员。这如下面的图 C.2 所示。由于作为祖（条目）直接下级的所有成员属于两个不同的结构客体类（comAddr 和postAddr），因此复合条目由两个家族组成，其中成员 1、成员 2 和成员 3 构成一个家族，成员 1、成员 4、成员 5、成员 6、成员 7、成员 8 和成员 9 构成另一个家族。

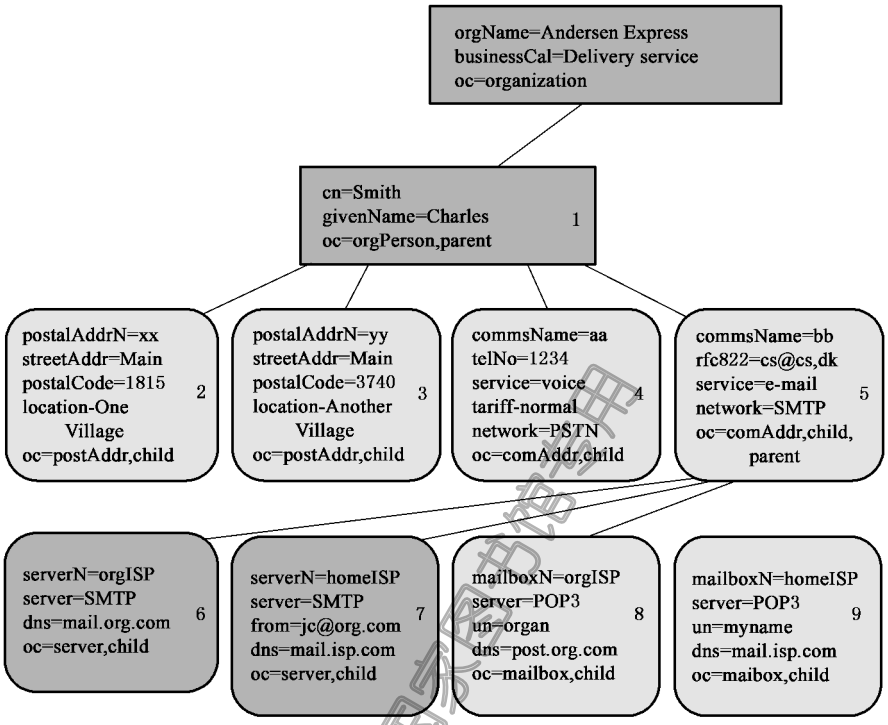

图 C.2 Charles Smith 条目家族

C.2.1 过滤器举例 1

现假设 Search 请求由{…o＝Andersen Express}的基本客体、{telNo＝1234 & service＝e-mail &streetAddr＝Main & postalCode＝3740}的过滤器、wholeSubtree 或oneLeve 的子集产生。当family-Grouping 参数设为以下值时：

a) entryOnly：复合条目中将没有任何单个成员匹配于过滤器；

b) strands：家族中将没有任何单个束匹配于过滤器；

c) multiStrand：每个家族中将没有任何束的组合或单个束匹配于过滤器；

d) compoundEntry：成员 2、成员 3、成员 4 和成员 5 将一起匹配于过滤器，并将被标记为贡献成员。所有的成员都将被标记为参与成员。

对上面情况 a)、情况 b)和情况 c)，将不返回任何本复合条目中的内容；

对上面情况 d)，返回的信息将依赖于家族返回规范：

1) contributingEntriesOnly：各成员标记为贡献成员，即成员 2、成员 3、成员 4 和成员 5 将被返回；

2) participatingEntriesOnly 和compoundEntry：复合条目中的所有成员都将被返回。

C.2.2 过滤器举例 2

如果将过滤器改为{rfc822＝cs@cs. dk & service＝e-mail & streetAddr＝Main & postalCode＝1815}，当familyGrouping 参数设为以下值时：

a) entryOnly:复合条目中将没有任何单个成员匹配于过滤器;

b) strands:任何家族中将没有任何单个束匹配于过滤器;

c) multiStrand:在成员 2 中结束的束以及任何通过成员 5 的束将匹配于过滤器。成员 2 和成员 5 有助于匹配,并将被标记为贡献成员。成员 1、成员 2、成员 5、成员 6、成员 7、成员 8 和成员 9 将被标记为参与成员;

d) compoundEntry:成员 2 和成员 5 将一起匹配于过滤器,并将被标记为贡献成员。所有的成员都将被标记为参与成员。

对上面情况 a)和情况 b),将不返回任何本复合条目中的内容。

对上面情况 c),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:各成员标记为贡献成员,即成员 2 和成员 5 将被返回;

2) participatingEntriesOnly:标记为参员成员的各成员将被返回,即成员 1、成员 2、成员 5、成员 6、成员 7、成员 8 和成员 9;

3) compoundEntry:复合条目中的所有成员都将被返回。

对上面情况 d),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:各成员标记为贡献成员,即成员 2 和成员 5 将被返回;

2) participatingEntriesOnly 和compoundEntry:复合条目中的所有成员都将被返回。

C.2.3 过滤器举例 3

如果将过滤器改为{rfc822=cs@cs.dk & service=e-mail},当 familyGrouping 参数设为以下值时:

a) entryOnly:只有成员 5 将匹配于过滤器,该成员将被标记为贡献成员和参与成员。

b) strands:任何通过成员 5 的束将匹配于过滤器。成员 5 将被标记为贡献成员。成员 1、成员 5、成员 6、成员 7、成员 8 和成员 9 将被标记为参与成员。

c) multiStrand:任何通过成员 5 的束以及任何邮政地址家族的束都将匹配于过滤器。成员 5 将被标记为贡献成员。成员 1、成员 2、成员 3、成员 5、成员 6、成员 7、成员 8 和成员 9 将被标记为参与成员。

d) compoundEntry:成员 5 将匹配于过滤器,并将被标记为贡献成员。所有的成员都将被标记为参与成员。

对上面情况 a),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly 和participatingEntriesOnly:成员 5 将被返回;

2) compoundEntry:复合条目中的所有成员都将被返回。

对上面情况 b),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:成员 5 将被返回;

2) participatingEntriesOnly:标记为参与成员的所有成员都将被返回,即成员 1、成员 5、成员 6、成员 7、成员 8 和成员 9;

3) compoundEntry:复合条目中的所有成员都将被返回。

对上面情况 c),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:成员 5 将被返回;

2) participatingEntriesOnly:标记为参与成员的所有成员都将被返回,即成员 1、成员 2、成员 3、成员 5、成员 6、成员 7、成员 8 和成员 9;

3) compoundEntry:复合条目中的所有成员都将被返回。

对上面情况 d),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:成员 5 将被返回;

2) participatingEntriesOnly 和compoundEntry:复合条目中的所有成员都将被返回。

C.2.4 过滤器举例4

如果将过滤器改为{cn=Smith & givenName=Charles}。只有祖(条目)将匹配于过滤器:

a) entryOnly:只有祖(条目)(成员1)将被标记为贡献成员和参与成员。

b) strands、multiStrand和compoundEntry:祖(条目)将被标记为贡献成员,所有的成员都将被标记为参与成员。

对上面情况a),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly和participatingEntriesOnly:成员1将被返回;

2) compoundEntry:复合条目中的所有成员都将被返回。

对上面情况b),返回的信息将依赖于家族返回规范:

1) contributingEntriesOnly:成员1将被返回;

2) participatingEntriesOnly和compoundEntry:复合条目中的所有成员都将被返回。

GB/T 16264.3—2008/ISO/IEC 9594-3:2005

中 华 人 民 共 和 国
国 家 标 准
信息技术 开放系统互连 目录
第3部分:抽象服务定义
GB/T 16264.3—2008/ISO/IEC 9594-3:2005
*
中国标准出版社出版发行
北京复兴门外三里河北街16号
邮政编码:100045
网址 www.spc.net.cn
电话:68523946 68517548
中国标准出版社秦皇岛印刷厂印刷
各地新华书店经销
*
开本 880×1230 1/16 印张 7 字数 204 千字
2008年12月第一版 2008年12月第一次印刷
*
书号:155066·1-34753

GB/T 16264.3-2008