LDAP 基础知识

天 津 南 开 创 元 信 息 技 术 有 限 公 司

Tianjin NanKai University Chuangyuan Information Technologies Co., Ltd.

ITEC 版权所有(©2000-2004)

天津总公司：天津市华苑产业区榕苑路 1 号软件出口基地 A 座 9 层
电　　话：022-83719090
传　　真：022-83719091
北京分公司：北京市海淀区昆明湖南路 62 号世纪城 6 区 2 号 18C
电　　话：010-88599909
传　　真：010-88596917

http://www.itec.com.cn　　E-mail:info@itec.com.cn

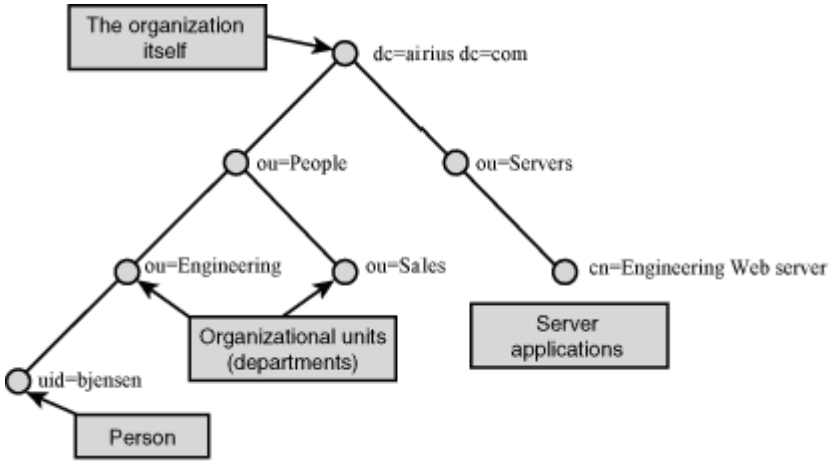
目 录

LDAP 基础知识

第一章 信息模型

LDAP 信息模型定义能够在目录中存储的数据类型和基本的信息单位。

目录的基本信息单元是条目(entry)——即关于对象的信息集合。通常，条目中的信息说明真实世界的对象，比如一个人。如果你看到一个典型的目录，就会发现数以千计的条目与人、部门、服务器、打印机等组织中的真实对象相符合。下图表示了典型的目录结构的一部分，其中的对象与组织中的真实世界对象相符。



条目由属性集合组成，每个属性说明对象的一个特征。每个属性有一个类型和一个或多个值。属性类型说明包含在此属性中的信息的类型，而值包含实际的数据。例如，下图表示的是描述人的一个条目，具有人的全名、姓、电话号码、电子邮件地址等属性。

属性类型	属性值
cn:	Zhang San
sn:	Zhang
telephoneNumber	+22 88832221
Mail	Zhangsan@itec.com

1.1 LDIF

本文中会看到很多用 LDIF 文本格式表示的目录条目，这是一种用文本格式表示目录的标准方式，可以用来从目录服务器导出数据或者向目录服务器导入数据。LDIF 文件只包含 ASCII 文本，从而允许通过电子邮件系统进行传输。因为它基于文本并且易读，所以使用 LDIF 代表目录条目。下面是一个用 LDIF 表示的目录条目：

```
dn: uid=bjensen, dc=airius, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Barbara Jensen
cn: Babs Jensen
sn: Jensen
mail: bjensen@airius.com
telephoneNumber: +1 408 555 1212
```

description: A big sailing fan.

一个LDIF条目由多行组成。第一行以dn:开始，后跟条目的分辨名。随后是条目的属性，每个属性一行。每个属性值前面是属性类型和冒号（:）。属性值的顺序无关紧要，但是为了使它更容易读，一般把objectClass属性放在第一个，并且把相同的属性类型的值放在一起。

有一种格式更复杂的 LDIF，用来完成对目录条目的修改。

1.2 模式(Schemas)

所有条目都有一个必须属性集合和一个可选属性集合。例如一个描述人的条目必须有一个 cn（通用名）属性和一个 sn（姓）属性。人的条目中有许多其他属性是可选的，而不是必须的。任何属性，如果没有明显说明是必须属性或可选属性，应该被禁止。

所有关于必须属性和可选属性的信息集合统称为模式(Schema)，目录模式对存储在目录中的信息的类型和值保持控制和维护。

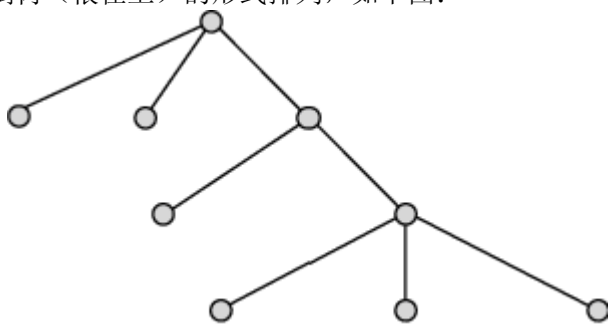
总之，LDAP 信息模型描述条目，条目是目录的基本信息单元。条目由属性组成，属性由一个属性类型和一个或多个属性值组成。属性的约束用来限制作为属性值的数据的类型和长度。目录模式规定了一个属性是必须属性还是可选属性。

第二章 命名模型

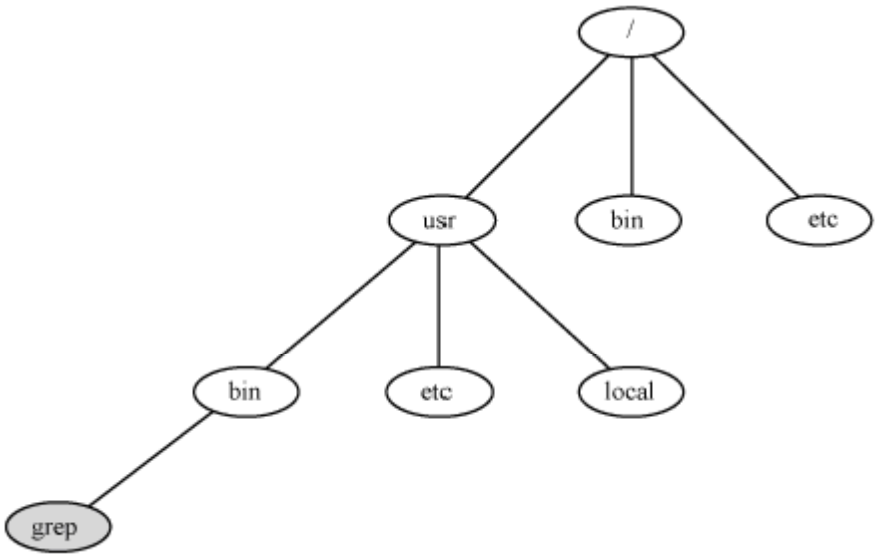
2.1 命名模型

LDAP 命名模型定义用户如何组织和引用数据。LDAP 命名模型提供的灵活性，使用户可以用一种易于管理的方式把条目放入目录。例如，可以创建一个条目，其中保存描述组织中人的信息的所有条目。或者选择与组织结构相符的方式

LDAP 模型指定条目以倒树（根在上）的形式排列，如下图：



熟悉 UNIX 文件系统读者会注意到它与目录结构的相似性（Windows 的目录结构也差不多）。这样的文件系统由一系列目录和文件组成每个目录由零个或多个文件或目录组成。下图是典型的 UNIX 文件系统的一部分：



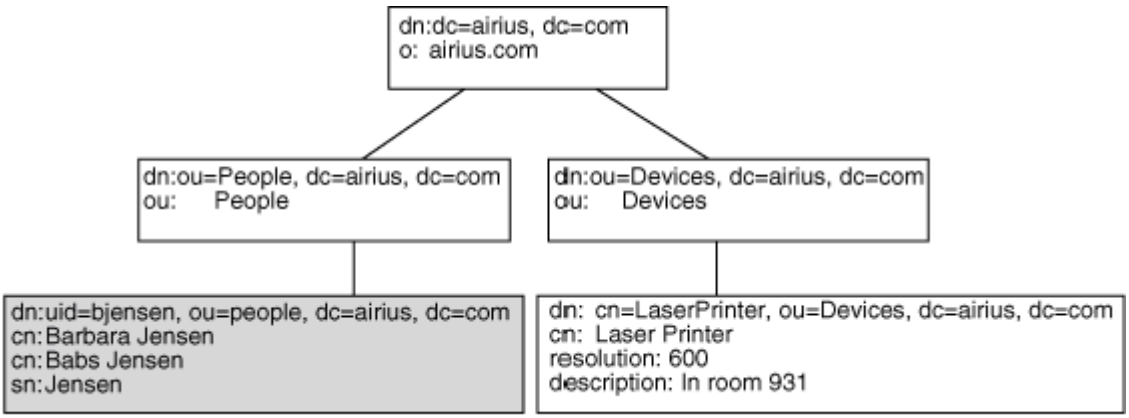
UNIX 文件系统和 LDAP 目录之间有三点不同：

第一是 LDAP 模型没有真正的根条目。文件系统有一个根目录，它是所有文件和目录的祖先。另一方面，在 LDAP 目录中，根条目是概念性的——不作为一个可以存放数据的条目存在。有一个称为 rootDSE 的条目，包含服务器相关的信息，但不是一个通常的目录条目。

第二是目录的每个节点都包含数据，任何节点都可能作为一个容器，即LDAP条目允许在他的下面有子节点。文件系统的每个节点或者是文件或者是目录，但不能同时是二者。在文件系统中，只有目录可以有子节点，而且只有文件可以包含数据。下图的目录树表示了此概念，注意条目dc=airius, dc=com, ou=People, 和 ou=Devices都包含数据（属性），也包含子节点。

文件系统和LDAP目录的第三个不同是树中的独立节点如何命名。LDAP名称与文件系统名称的方向是相反的。图示这一点。让我们看看上图和下图中的加阴影的节点。上图中，加阴影的节点是完整名称是/usr/bin/grep的文件。注意，如果你从左向右读文件名，你是从树的顶端(/)到命名的文件。

与之相对的是下图中加阴影的目录条目的名称。它的名称是uid=bjensen, ou=people, dc=airius, dc=com。注意，如果你从左向右读，你是从命名的条目逆向到达树的顶端。



正如你看到的，LDAP 支持目录条目的层次排列。但并不是强制某种类型的层次。正如可以按照你喜欢和方便的方式自由地排列文件，你也可以自由地构造你希望的目录层次。当然，根据你的情况，一些目录结构要好些。

这种自由的一个例外是当你的 LDAP 目录服务实际是一个 X.500 服务的前端。X.500 命名模型比 LDAP 严格得多。在 X.500 1993 标准中，目录结构规则（directory structure rules）限定你能创建的层次类型。标准通过指定什么类型的对象类能作一个条目的直接子条目。比如，在 X.500 模型中，只有描述国家（countries）、地点（localities）和组织

（organizations）的条目才能成为目录树的根。LDAP 命名模型不对树结构作任何限定，任何类型的条目可被放置在树的任何位置。

除指定你如何排列目录条目的层次结构，LDAP 命名模型还描述了你如何在指向目录中的独立条目。下文将详细说明。

2.2 命名为什么重要？

命名模型是为目录中的每个条目给出一个唯一的名称。在 LDAP 中，使用分辨名(DN)来唯一标识条目。

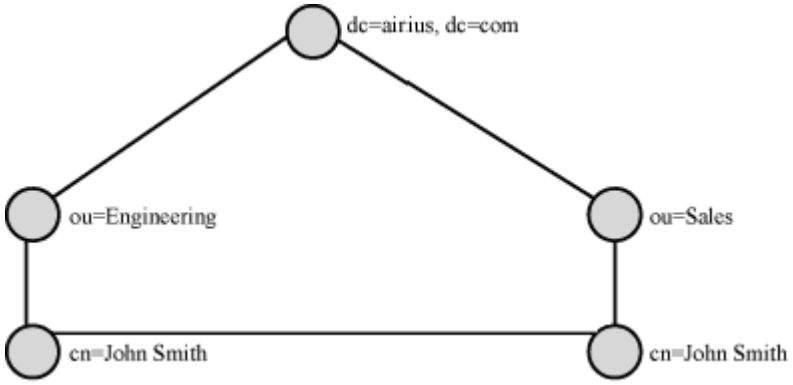
类似于文件系统的路径名，LDAP条目的名称是其父条目到根的独立名称连接排列而成。比如，看看上图中的目录树。加阴影的条目的名称是uid=bjensen, ou=people, dc=airius, dc=com。从左向右读条目名称，你能跟踪到从条目本身到目录树根的路径。名称的各独立部分以逗号分隔。逗号后的空格是可选的，以下两个分辨名是等价的：

uid=bjensen, ou=people, dc=airius, dc=com

uid=bjensen,ou=people,dc=airius,dc=com

在任何条目的 DN 中，最左边的一段被称为相对分辨名(RDN)。在兄弟条目间(他们拥有共同的直接父条目)，每个 RDN 必须是唯一的。由于这个规则在条目树中递归有效，所以没有两个条目有相同的 DN。如果你尝试添加两个有相同名字的条目，目录服务器将拒绝你的第二个条目，这和 UNIX 很相似。

注意：只有拥有共同直接父条目的RDN必须唯一。下图中，虽然有两个条目的RDN都是 cn=John Smith，但他们在不同的子树，所以是合法的。这种构造目录的方式好不好就是另一回事了。



2.3 凌乱的 RDN 主题：多值 RDN 和引证

你可能已经注意到了上述的每个 RDN 都是由两部分组成：一个属性名称和一个值，以等号(=)分隔。RDN 也可以包含多于一个的名称/值对，被称为多值 RDN，象下面这样：

cn=John Smith + mail=jsmith@airius.com

这个条目的RDN由两个“属性=值”对组成：cn=John Smith 和 mail=jsmith@airius.com。

多值 RDN 可用来在其他都相同的时候区分 RDN。例如，如果同一个容量中有多于一个的 John Smith，多值 RDN 允许你为每个条目指定唯一的 RDN。无论如何，你应该避免在目录中使用多值 RDN。他会使你的命名空间趋向混乱，最好是为每个条目取唯一的名称。

既然DN中的RDN是用逗号分割的，你可能想知道如果RDN中包含逗号如何表示。如何区别哪个是RDN包含的逗号，哪个是分隔RDN的逗号呢？比如，如何理解一个名为o=United Widgets,Ltd.的条目？

如果在你的目录中有这样的 DN，你必须用反斜杠转义表示全部文字上的逗号（那些 RDN 中的逗号）。在我们的例子中，DN 将被表示为：o=United Widgets\, Ltd., c=GB

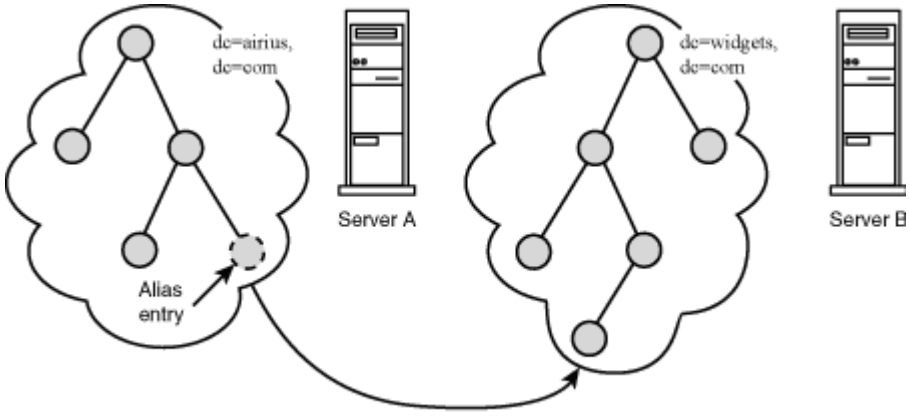
其他几个特定字符在出现 DN 中的时候，也必须转义。下表列出根据 LDAPv3 必须转义的全部字符：

DN 或 RDN 开头或结尾的空格	32	\<空格>
DN 或 RDN 开头或结尾的井号(#)	35	\#

逗号(,)	44	\,
加号(+)	43	\+
双引号(")	34	\"
反斜杠(\)	92	\\
小于号(<)	60	\<
大于号(>)	62	\>
分号(;)	59	\;

2.4 别名（Aliases）

LDAP 目录中的别名条目允许一个条目指向另一个条目，意味着你可以设计非严格层次关系的目录结构。别名条目的功能相当于 UNIX 的符号链接（symbolic links）或 Windows 95/NT 的快捷方式。下图中，标出的条目指向“真正”的条目。



在目录中创建一个别名条目，你必须首先创建一个对象类是alias并且拥有属性aliasedObjectName的条目。属性aliasedObjectName的值必须是别名指向的条目的DN。

不是所有的 LDAP 目录服务器都支持别名。因为别名可以指向任何目录条目，甚至是在不同服务器上的条目，所以别名会影响性能。考虑上图中的目录树，别名条目在一个目录树中，指向在另一服务器上的另一目录树。为支持跨越目录树的查询，服务器 A 必须在进行查询操作中一遇到别名条目就与服务器 B 联系。这将显著地降低查询速度，这就是某些目录软件不支持别名的主要原因。ITEC-iDS 3.3 支持别名。

通常，你使用别名所要达到的目标都可以通过引用（referrals）达到，或者在条目中放置 LDAP URLs 使客户端能够找到指向的信息。

第三章 功能模型和 API 使用

我们需要一种对目录树中的信息进行访问的方法。LDAP 功能模型说明了能够使用 LDAP 协议对目录执行的操作。

LDAP 功能模型包含一个可以分成三组的操作集合。查询类操作允许用户搜索目录并取回目录数据。更新类操作允许用户对目录条目进行添加、删除和修改。认证和控制类操作允许客户端向目录证明自己的身份，并在几个方面对会话进行控制。

除了这三类主要的操作，LDAPv3 协议通过 LDAP 扩展操作定义了添加新操作的框架。扩展操作以一种有序的方式对协议进行扩展来满足新的市场需要。

《RFC1823 LDAP 应用程序接口》定义了 LDAP 的 C API 接口。IEFT 组织正在制定 LDAP 的 JAVA API 接口标准，到本文截稿时止，尚未形成正式的 RFC,最新的草案是 2002 年 6 月公布的 draft-ietf-ldapext-ldap-java-api-18.txt，即草案第 18 稿。下文基本是根据 RFC1823，其他语言的 API 接口是相似的，可以参考。

3.1 LDAP 查询类操作

查询类操作允许用户搜索目录并取回目录数据，有两个查询操作：查询和比较。

3.1.1 查询操作

LDAP 查询操作用来在目录中搜索条目，并取出单个目录条目。LDAP 没有读操作，当需要读取某条目时，必须使用一种特殊格式的查询操作，其中限定了你要取回的条目内容。

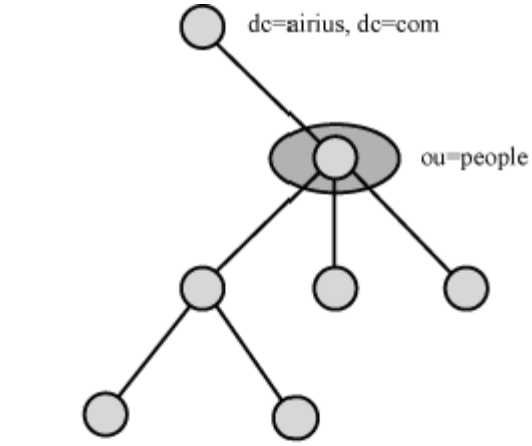
```
int ldap_search(
                LDAP    *ld,
                char    *base,
                int     scope,
                char    *filter,
                char    *attrs[],
                int     attrsonly
);
```

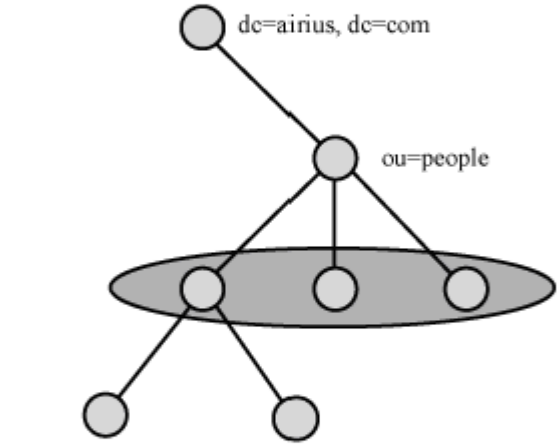
LDAP 查询操作需要 6 个参数。

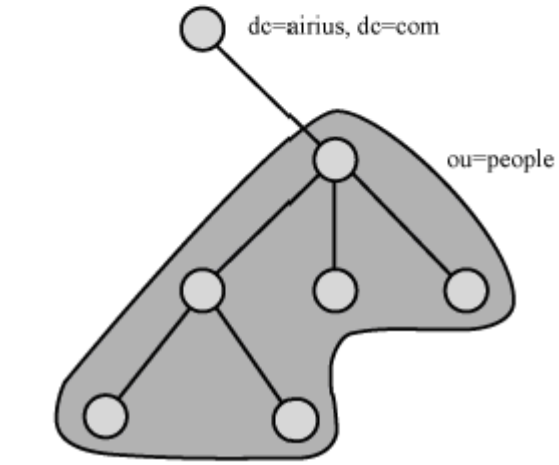
第一个参数是 LDAP 连接的句柄。

第二个参数是查询的基对象，这个参数用一个 DN 来表示，它表示你要查询的目录（子）树的根。

第三个参数是范围。范围有三个类型。一是BASE，是指你要限定查询条件为基对象。这通常用来在目录中找一个特殊的条目。二是ONELEVEL，是指你想要查询的只是基对象直接下级的条目。三是SUBTREE，是指你想要查询从基对象以任何路径到树叶的整个子树。下图描述了查询范围的三种类型：







第四个查询参数是查询过滤条件。这是一个描述返回条目类型的表达式。过滤表达式在 LDAP 查询操作中使用非常灵活，参见本文的“第七章 LDAP 查询过滤”部分。

第五个查询参数是查询结果返回的属性列表。你可以指定所有的属性都要被返回，也可要求返回条目的某几个属性。

第六个查询参数是attrsonly。这是一个布尔参数，如果它被设置为真的话，服务器将只把属性类型传送给客户端，而不传送属性值。当客户端只想知道条目所包含的属性而不想知道实际的

值时，这个参数将会用到。如果这个参数被设置为假，属性类型和属性值都将被返回。

3.1.2 比较操作

另一种查询类操作是比较操作，它用于检查某条目是否包含某个属性值。如果条目有此值，则比较结果为真，否则，比较结果为假。

看上去，比较操作是重复的。毕竟，如果你想要确定一个特殊条目是否包含一个特殊的属性值，你仅需执行一个查询基础等于指定目录的 DN，一个基础范围，一个想要的测试的过滤表示的查询。如果有条目被返回，则测试成功，否则，测试失败。

存在比较操作是由于历史和与 LDAP 起源于 X.500 有关。在比较和查询操作行为中仅仅有一点不同。如果企图在一个属性上进行比较，但这个属性并不存在于目录中，比较操作将为客户端返回一个特殊的指示——属性不存在。而查询操作则简单的不返回条目。区别“条目有属性但是没有匹配值”和“条目根本没有属性”的能力在某些情况下会方便一些。比较操作的另一优势是服务器和客户机间交换的数据量更小。

3.2 LDAP 更新类操作

LDAP 更新类操作包括四种操作：添加、删除、修改和重命名（即修改 RDN），这四种操作定义了在目录中操作数据的方式。

3.2.1 添加操作

```
int ldap_add( LDAP *ld, char *dn, LDAPMod *attrs[] );
```

添加操作创建新的目录条目，它的后两个参数：要创建的条目的分辨名 DN 和新条目中包含的属性/属性值对的集合。为了使添加操作成功，必须满足以下四个前提条件：

- 新条目的父条目必须已经存在。
- 不能存在同名（分辨名）的条目
- 新条目必须与有效的模式(schema)相一致。
- 访问控制必须允许执行此操作。

3.2.2 删除操作

```
int ldap_delete( LDAP *ld, char *dn );
```

删除操作只需指明要删除的条目 DN。只能删除目录的叶节点，即不支持删除子树。删除一经进行，无法恢复。为了使删除操作成功，必须满足以下两个前提条件：

- 新条目的父条目必须已经存在。
- 访问控制必须允许执行此操作。

3.2.3 修改操作

```
int ldap_modify( LDAP *ld, char *dn, LDAPMod *mods[] );
```

修改操作除指明操作的条目 DN 外，还包括 LDAPMod 结构数组。其中每个数组元素是一个修改动作，记录修改的操作类型和操作数据。修改操作功能强大，能完成对条目的属性类型和属性值的修改操作。

3.2.4 重命名（修改 RDN）操作

重命名，或者修改 RDN 操作，用于为条目重命名。

```
int ldap_modrdn(
                LDAP    *ld,
                char    *dn,
                char    *newrdn,
                int     deleteoldrdn
    );
```

它有四个参数：LDAP 连接的句柄、要重命名的条目、条目新的 RDN 和删除原 RDN 标志(delete-old-RDN)。为了使修改 RDN 操作成功，必须满足如下前提条件：

- 被重命名的条目必须已经存在。

- 条目的新名称不能已经被其他条目使用。
- 访问控制必须允许执行此操作。

3.3 认证和控制类操作

认证和控制类操作包括两个 LDAP 认证操作——绑定和解绑定，一个控制操作——放弃。

3.3.1 绑定操作

一般常用的是使用简单密码的绑定。使用 SSL/TLS 绑定更加复杂，一般需要预先配置和多步操作，目前 RFC 没有对此作出规定。

3.3.2 解绑定操作

用于中断持续进行的 LDAP 操作，关闭连接。

3.3.3 放弃操作

放弃操作用于中断正在进行的操作。

第四章 安全模型

4.1 安全模型

4.1.1 简介

在计算机组成的网络世界中安全性极其重要，同样在 LDAP 中也是如此。当通过不安全的网络向内或向外传送数据时，需要对敏感信息进行保护。同时，需要知道谁在请求信息和谁在发送信息，特别是对目录执行更新操作时，这一点更重要。"安全性"这一术语通常包含四个方面：

- 认证。保证对方（机器或人）的身份与所声明的相符。
- 完整性。保证到达的信息与发送的信息相同。
- 保密性。使用数据加密手段对要暴露的信息进行保护。
- 授权。保证某一团体所请求的操作确实是被允许的。这通常在认证之后进行检查。

LDAPv3 是一个功能强大的目录访问协议，它定义了一组安全功能，在所有的 LDAP 客户机和服务器之间保持这些功能的互操作性非常重要。

对目录服务的安全威胁主要来自如下几方面：

（1） 使用读取操作对目录数据进行未授权的访问。
（2） 通过监视他人的访问对客户机的认证信息进行未授权的访问。
（3） 通过监视他人的访问对目录数据进行未授权的访问。
（4） 对目录数据进行未授权的修改。
（5） 对配置信息进行未授权的修改。
（6） 对资源进行未授权的或超量的使用。
（7） 对目录进行 IP 地址欺骗。通过修改传输的数据或误导客户机的连接，使客户机相信信息来自目录，而事实上并不是。

(1) (4) (5) (6)来自有敌意的客户机，(2) (3) (7) 来自客户机和服务器之间的路径中有敌意的代理，或伪装的服务器。

LDAP 协议族提供了如下安全机制对目录服务进行保护：

（1） 使用请求者 ID 对客户机进行认证。
（2） 通过 SSL/TLS 协议来提供数据完整性保护。
（3） 通过 SSL/TLS 协议来提供数据保密性保护。
（4） 通过对服务器的管理员权限进行配置和对服务器资源的使用进行限制。
（5） 通过 SSL/TLS 协议对服务器进行认证。

在 LDAP 之外，访问控制也已经成形。

4.1.2 需要的安全机制

（1） 在只读的公共目录中，使用匿名认证。

（2） 对于需要数据安全性（包括完整性和保密性）和认证的目录，可以使用 StartTLS 和简单认证。

4.1.3 LDAP 绑定操作

安全模型基于绑定操作。有几种不同的可选绑定操作，因此所应用的安全也不同。一种可能性是客户进程提供分辨名 DN 和一个明文密码请求访问目录。如果没有分辨名 DN 和密码，服务器就假定是一个匿名会话。强烈建议不要使用明文密码，因为基础的传输服务不能保证机密性，从而可能导致把口令暴露给未授权的团体。

LDAP 版本 3 中可以使用扩展协议操作。一个与安全有关的扩展是《RFC2830 轻量级目录访问协议(v3):传输层安全扩展》。它定义了几个操作，这些操作使用 TLS 方法对 LDAP 会话进行加密保护其免受欺骗。TLS 在《RFC2246 TLS 协议版本 1.0》中定义。它基于网景通讯公司安全套接层协议 SSL 3.0。TLS 有一个能与 SSL 服务器通讯的机制，以便于它保持向后兼容。SSL 和 TLS 的基本原理相同。

一旦客户进程被识别，就可以查询出访问控制信息，并用它来确定客户进程是否有足够的权限去做所请求的操作。

4.1.3.1 无认证(匿名)

这是最简单的一种方法，显然不需要太多的解释，这种方法只在没有数据安全问题并且不涉及访问控制权限的时候才能使用。例如，当你的目录是任何人都可以浏览的地址簿时，就是这种情况。当你调用 API 的绑定操作时分辨名(DN)和密码保留为空，目录会假定是无认证，LDAP 服务器会自动假定一个匿名用户会话并且服务相应的访问控制权限。

4.1.3.2 基本认证(简单密码)

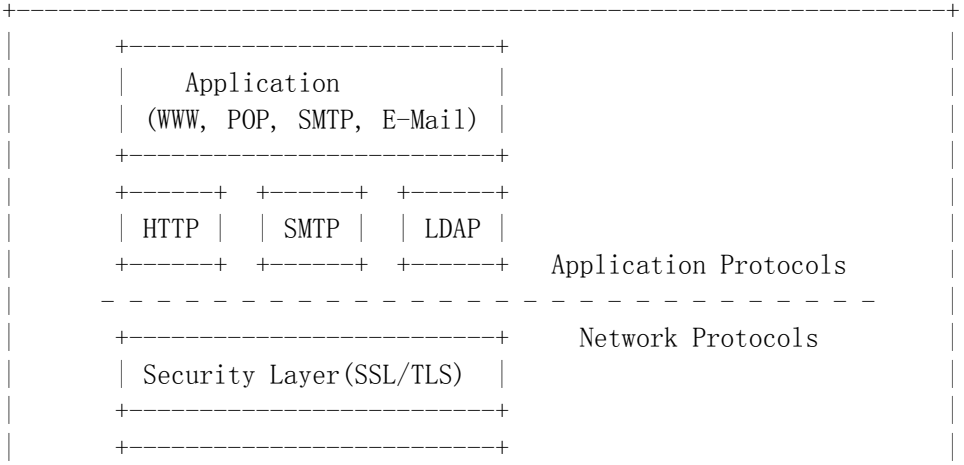
当 LDAP 客户进程和服务进程之间建立连接时，会协商安全机制，这是在 LDAP 应用程序接口(API)中指定的方法。除了根本不使用认证之外，最简单的 LDAP 安全机制是基本认证，它也用于其它 Web 相关的协议，如 HTTP。

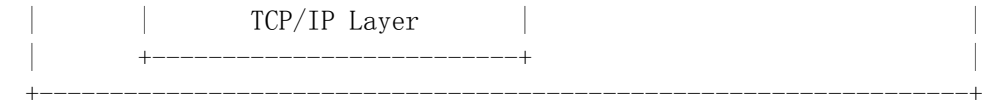
当使用 LDAP 的基本安全认证时，客户进程通过网络向服务进程发送一个分辨名(DN)和口令来标识自己（有的实现中使用 Base64 编码）。服务进程检查客户进程发送的分辨名(DN)和密码是否与目录中存储的分辨名(DN)和密码相匹配，如果匹配则认为通过了认证。Base64 编码在《RFC1521 多用途因特网邮件扩展(MIME)标准》中定义。与其说它是一种加密不如说它是一种编码，因此一旦某人在网络中得到此数据，破译它并不难。

4.1.3.3 SSL/TLS

安全套接层协议的意图是提供认证和数据的安全性，它封装了 TCP/IP 套接字，以便于每个 TCP/IP 应用都能够使用它保证通讯的安全。

SSL/TLS 与其他协议的关系见下图：



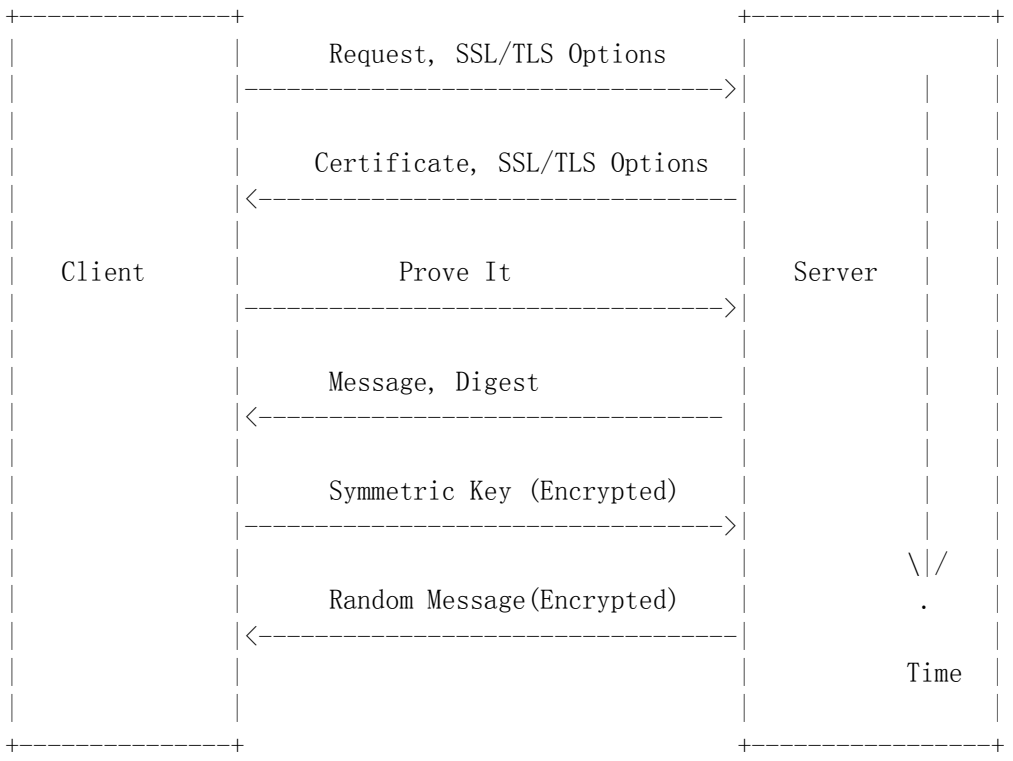


SSL 由 Netscape 开发，当前版本是 3.0，传输层安全性(TLS)是一个正在发展中的标准，由 IETF 制定，它基于 SSL3.0，且差别甚微，它假定 TLS 将取代 SSL，且与 SSL3.0 向后兼容。下面的讨论适用于 SSL 和 TLS 两者。

SSL/TLS 支持服务器认证（客户机确认服务器），客户机认证（服务器认证客户机）、或双向认证，它对网络中发送的数据进行加密来提供保密性。

SSL/TLS 使用公共密钥方法保证通讯和认证双方的安全性，有一对公钥/私钥。他们彼此是互逆操作，这意味着使用私钥加密的数据可以使用公钥来解密，反之亦然。

客户进程和服务进程之间协商 SSL/TLS 连接的最简单的交换如下图所示



1. 客户进程向服务进程请求 SSL/TLS 会话，在请求中也包括客户进程所支持的 SSL/TLS 选项。
2. 服务进程发回它的选项和证书，包括服务器的公钥，证书使用者的 ID（如分辨名），证书名和有效期，可以把证书看成电子护照，它必须由受信任的权威机构来发行，它保证公钥确实属于证书中所提到的实体。发证者签名的证书可以通过可免费获得的公钥来验证。
3. 客户进程请求服务进程证明它的身份，即确定证书不是截获此证书的其他人发送的。
4. 服务器发回一个包括 message digest 用私钥加密的消息。

4.2 与目录安全性相关的协议

4.2.1 关于安全的核心协议：

- RFC2820，Access Control Requirements for LDAP。描述用于 LDAP 目录服务的 ACL 模型的基本需求。提供对目录的授权访问和目录间互操作性所需要的访问控制的需求。
- RFC2829， Authentication Methods for LDAP 。描述 LDAP 认证方法。
- RFC2830，Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security。为 LDAPv3 定义了启动传输层安全性操作（StartTLS），此操作使用 LDAP

扩展请求来定义。

4.2.2 关于 LDAP 的最新进展。

4.2.2.1 LDAP 修订工作组

Internet-Drafts:

- Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names (20004 bytes)
- Lightweight Directory Access Protocol (v3) (142805 bytes)
- Lightweight Directory Access Protocol (v3):Technical Specification (10770 bytes)
- The String Representation of LDAP Search Filters (20906 bytes)
- Authentication Methods and Connection Level Security Mechanism for LDAPv3 (94536 bytes)
- The LDAP URL Format (25344 bytes)
- IANA Considerations for LDAP (42339 bytes)
- A Summary of the X.500(3rd edition) User Schema for use with LDAPv3 (45558 bytes)
- Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions (103450 bytes)

4.2.2.2 LDAP 扩展操作

Internet-Drafts:

- The Java LDAP Application Program Interface(271414 bytes)
- LDAP Extensions for Scrolling View Browsing of Search Results(26038 bytes)
- Access Control Model for LDAP (133951 bytes)
- X.509 Authentication SASL Mechanism (20431 bytes)
- LDAP Control for a Duplicate Entry Representation of Search Results (19779 bytes)
- A Taxonomoy of Methods for LDAP Clients Finding Servers (10828 bytes)
- Discovering LDAP Services with DNS (13272 bytes)
- Lightweight Directory Access Protocol over UDP/IP (12857 bytes)

RFC:

- Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services (RFC 2589) (26855 bytes)
- Use of Language Codes in LDAP (RFC 2596) (17413 bytes)
- An LDAP Control and Schema for Holding Operation Signatures (RFC 2649) (20470 bytes)
- LDAP Control Extension for Simple Paged Results Manipulation (RFC 2696) (12809 bytes)
- Access Control Requirements for LDAP (RFC 2820) (18172 bytes)
- Authentication Methods for LDAP (RFC 2829) (33471 bytes)
- Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security (RFC 2830) (24469 bytes)
- LDAP Control Extension for Server Side Sorting of Search Results (RFC 2891) (15833 bytes)

第五章 schema

5.1 schema 的目的

目录的模式(schema)是一组规则,用来确定目录能存储什么和服务器和客户端在进行目录操作时如何对待数据. schema 被 LDAP 各方(服务器、客户端、应用程序)用来作为数据交换的"标准",以保证对数据理解的正确性,从而保证了 LDAP 协议的开放性.

schema 也被用作目录中存储的数据的长度、范围和格式的强制约束.

最后,schema 也有利于目录中数据的规整和对访问者权限的控制.

5.2 schema 构成元素

构成 schema 的元素有属性类型(attribute types)、对象类(object classes)、语法(syntaxes)、匹配规则(matching rules)、目录信息树内容规则(DIT content rules)、目录信息树结构规则(DIT strctural rules)和命名形式(name forms).

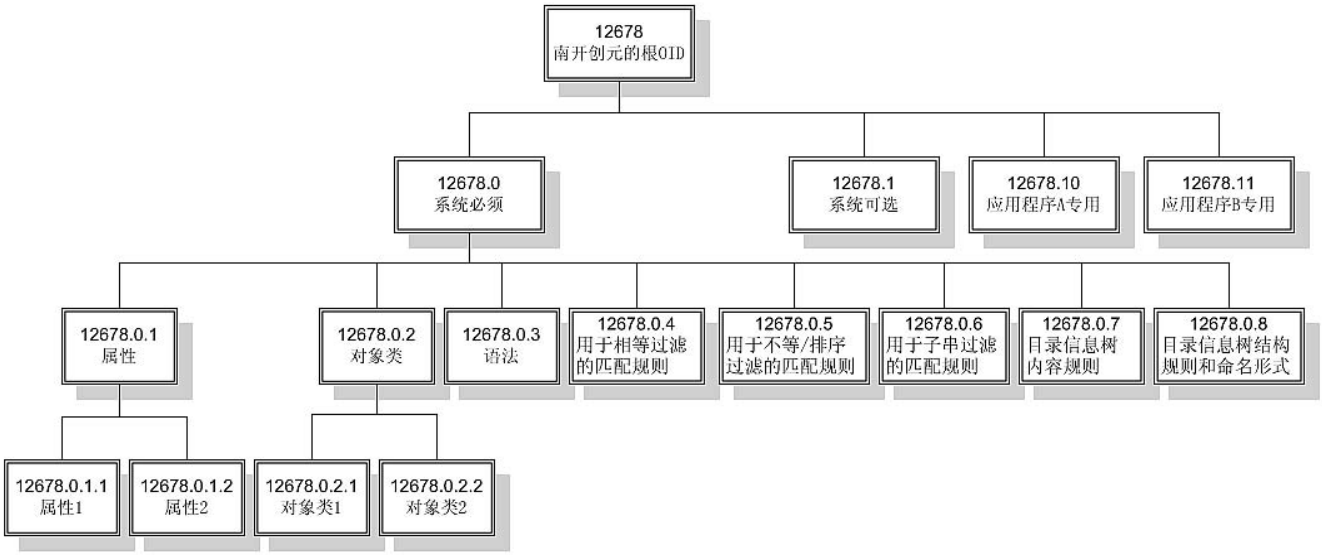
5.3 schema 准备知识

5.3.1 OID(object identifer,对象标志符)

每一个 schema 元素由一个 OID(对象标志符)来作唯一标志.不同用户自定义的 schema 元素名称可能冲突,但只要设置正确,OID 是全球唯一、不会冲突的. OID 是由 1 段或多段十进制数字组成,段之间以点(%x2E,".")分隔,例如"1.2.3.4". 其中第 1 段(或称根 OID,base OID)需向 IANA 免费申请.如果用户需要自定义 schema 要素,应申请自己组织的根 OID,不要使用其他组织的或臆想的根 OID,这可能导致非常隐蔽和费解的错误.

申请OID的IANA网址: http://www.iana.org/cgi-bin/enterprise.pl
IANA的OID清单网址: http://www.iana.org/assignments/enterprise-numbers
IANA的OID信息网址: http://www.alvestrand.no/harald/objectid/

OID 的分段是为了体现树状层次关系,即 1.2.3.4 是 1.2.3 的直接子类,1.2.3 是 1.2 的直接子类.下图是 schema 元素层次关系树示例:



5.3.2 schema 元素的名称

用户自定义的 schema 元素的名称由不同组织设置,没有机制能保证不发生冲突.所以,在本组织外或跨服务器使用自定义的 schema 元素时,应使用 OID 而不是名称.同时,在自定义的

schema 元素名称前冠以标志本组织的单词也是值得提倡的,至少有助于区分 schema 元素是由谁定义的.

5.4 schema 元素的格式

LDAPv3 中,schema 的构成是由《RFC2252 Lightwright Directory Access Protocol(v3): Attribute Syntax Definitions》(以下简称 RFC2252)定义的.IETF 为制定 LDAPv4 而形成的草案《LDAP: Directory Information Models <draft-ietf-idapbis-models-01.txt>》(以下简称"models 草案")对 RFC2252 作了部分修订.本文将主要以 RFC2252 为依据,同时兼顾草案的新内容.

schema 元素有系统定义的和用户定义两类.一般用户只能定义属性类型和对象类. 系统定义的 schema 元素最小集合由 RFC2252、models 草案和《RFC2256 A Summary of the X.500(96) User Schema for use with LDAPv3》(以下简称 RFC2256)规定,这是一般的 LDAP server 都支持的.除此之外,各 LDAP server 还会提供一些其他 RFC 定义的和生产商自行制定的属性类型和对象类, 同时支持用户自定义. 但提供的语法和匹配规则基本都是 RFC 规定的,极少增减.

5.4.1 属性类型(attribute types)

属性类型控制属性格式,包括属性的语法、匹配规则、是否可以多值、修改权限和用法等.属性类型可直接由 0 或多个属性类型继承而来,形成属性类型的层次关系树.

5.4.1.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
AttributeTypeDescription = LPAREN WSP
    numericoid                  ; OID
    [ SP "NAME" SP qdescrs ]    ; 名称
    [ SP "DESC" SP qdstring ]   ; 描述
    [ SP "OBSOLETE" ]           ; 废弃
    [ SP "SUP" SP oid ]         ; 父类
    [ SP "EQUALITY" SP oid ]    ; 用于相等过滤的匹配规则(equality filter
matching rule)
    [ SP "ORDERING" SP oid ]    ; 用于排序过滤的匹配规则(ordering filter
matching rule)
    [ SP "SUBSTR" SP oid ]      ; 用于子串过滤的匹配规则(substrings filter
matching rule)
    [ SP "SYNTAX" SP noidlen ] ; 值的语法
    [ SP "SINGLE-VALUE" ]       ; 是否单值(缺省多值)
    [ SP "COLLECTIVE" ]         ; 是否聚集(缺省非聚集),此选项必须与 USAGE
userApplications 同时设置
    [ SP "NO-USER-MODIFICATION" ] ; 是否允许用户修改(缺省允许用户修改),此选项与
USAGE userApplications 不能同时设置
    [ SP "USAGE" SP usage ]     ; 用法,缺省 userApplications
    extensions WSP RPAREN       ; 扩展
```

说明:

至少应定义 SUP 或 SYNTAX 之一.

语法后可选的[LCURLY len RCURLY]限定字符串属性值的字符数上限或其他属性值的字节数上限,但不是语法标识符的一部分.注意对于基于 UTF-8 的属性值,1 个字符占用 1 个或多个字节.

LDAP server 必须实现 RFC2252 的 5.1、5.2、5.3 描述的属性类型.

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

```
usage = "userApplications"     /     ;应用程序(用户)使用
        "directoryOperation"   /     ;目录系统操作
        "distributedOperation" /     ;DSA-shared
        "dsaOperation"               ;DSA-specific
```

用户属性(user attribute)的 usage 设为 userApplications.操作属性可设为 directoryOperation、distributedOperation 和 dsaOperation 之一.

操作属性(operational attribute) 包含目录用于在内部跟踪修改和子树属性的信息。除非明确请求，否则响应搜索时不会返回操作属性,应设为 NO-USER-MODIFICATION.

无论用户属性还是操作属性,都必须与父类的 usage 完全相同. 属性从父类继承匹配规则和语法.

5.4.2 对象类(object classes)

对象类是"共享某些特性的对象的识别家族",即对象的模板. 是通过定义条目中所含的属性来定义目录中的条目类型.

5.4.2.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
ObjectClassDescription = RPAREN WSP
        numericoid                 ; OID
        [ SP "NAME" SP qdescrs ]   ; 名称
        [ SP "DESC" SP qdstring ]  ; 描述
        [ SP "OBSOLETE" ]          ; 废弃
        [ SP "SUP" SP oids ]       ; 父类
        [ SP kind ]                ; 类型
        [ SP "MUST" SP oids ]      ; 必须的属性
        [ SP "MAY" SP oids ]       ; 可选的属性
        extensions WSP RPAREN      ; 扩展
```

说明:

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

kind = "ABSTRACT" / "STRUCTURAL" / "AUXILIARY",分别表示抽象类、结构类和辅助类. 缺省是 STRUCTURAL.

抽象类提供基础特性以供其它对象类继承. 抽象类只能继承自抽象类.结构类都是直接或间接继承自抽象类 top.辅助类可以没有父类. 单独的抽象类不能定义条目.

结构类用来定义 DIT 的结构.只有结构类能独立定义条目. 条目一经定义,其结构类不能改变. 结构类不能继承自辅助类.

辅助类用来增加条目的特性.通常被用来增加条目的必须和可选属性.辅助类不能继承自结构类.每个条目可以属于任意数量的辅助类,条目的辅助类可以在任何时候改变.

5.4.3 语法(syntaxes)

语法没有名称,所以在使用时应引用 OID.

5.4.3.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
SyntaxDescription = LPAREN WSP
    numericoid                 ; OID
    [ SP "DESC" SP qdstring ]  ; 描述
    extensions WSP RPAREN      ; 扩展
```

5.4.4 匹配规则(matching rules)

为服务器在搜索操作过程中如何比较字符串提供准则。在国际搜索中，匹配规则告知服务器所用的对照顺序及运算符

用途:

- ✧ server 执行查询或比较操作时比较属性值
- ✧ server 修改条目时确定要添加或删除的属性值
- ✧ server 在比较 DN 和条目名称时使用

匹配规则可分为用于相等过滤、用于不等/排序过滤、用于子串过滤和用于 subschema 属性的四类.大多数的属性都有用于相等过滤的匹配规则,但不一定有其他匹配规则.

5.4.4.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

匹配规则:

```
MatchingRuleUseDescription = LPAREN WSP
    numericoid                ; OID
    [ SP "NAME" SP qdescrs ]  ; 名称
    [ SP "DESC" SP qdstring ] ; 描述
    [ SP "OBSOLETE" ]         ; 废弃
    SP "SYNTAX" SP numericoid ; 语法 OID
    extensions WSP RPAREN     ; 扩展
```

说明:

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

匹配规则使用(matching rule use):

匹配规则使用列出适用于某个匹配规则的属性类型.但 LDAPv3 未对其进行更详细地说明,一般的 LDAP server 也未予实现. models 草案重新启用此概念,将其定义在 subschema subentry 中.在此仅作一般介绍,供用户了解 LDAP 协议的最新发展方向.

```
MatchingRuleUseDescription = LPAREN WSP
        numericoid                ; OID
        [ SP "NAME" SP qdescrs ]  ; 名称
        [ SP "DESC" SP qdstring ] ; 描述
        [ SP "OBSOLETE" ]         ; 废弃
        SP "APPLIES" SP oids      ; 属性类型列表,指明适用于本匹配规则的属性类
型
        extensions WSP RPAREN     ; 扩展
```

说明:

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

5.4.5 目录信息树内容规则(DIT content rules)

此概念在 LDAPv3 中没有提到,在 models 草案中被定义为"可以实现"的功能.目前还没有 LDAP server 支持此功能,在此仅作一般介绍,供用户了解 LDAP 协议的最新发展方向.

目录信息树内容规则是"支配特定结构对象类的条目的内容的规则".它针对属于特定对象类

的条目,用于指定附加其上的辅助类的哪些属性是必须、可选或禁止的.

5.4.5.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
DITContentRuleDescription = LPAREN WSP
    numericoid                  ; OID
    [ SP "NAME" SP qdescrs ]    ; 名称
    [ SP "DESC" SP qdstring ]   ; 描述
    [ SP "OBSOLETE" ]           ; 废弃
    [ SP "AUX" SP oids ]        ; 辅助对象类列表
    [ SP "MUST" SP oids ]       ; 必须的属性类型
    [ SP "MAY" SP oids ]        ; 可选的属性类型
    [ SP "NOT" SP oids ]        ; 禁止的属性类型
    extensions WSP RPAREN       ; 扩展
```

说明:

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

5.4.6 目录信息树结构规则(DIT strctural rules)

此概念在 LDAPv3 中没有提到,在 models 草案中被定义为"可以实现"的功能.目前还没有 LDAP server 支持此功能,在此仅作一般介绍,供用户了解 LDAP 协议的最新发展方向.

目录信息树结构规则是"通过指定允许的父类和下级条目的关系来支配 DIT 结构的规则. 1 个目录信息树结构规则与 1 个命名形式相联系."

5.4.6.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
DITStructureRuleDescription = LPAREN WSP
    ruleid                      ; 规则 ID
    [ SP "NAME" SP qdescrs ]    ; 名称
    [ SP "DESC" SP qdstring ]   ; 描述
    [ SP "OBSOLETE" ]           ; 废弃
    SP "FORM" SP oid            ; 命名形式(name forms),参见下一小节.
    [ SP "SUP" ruleids ]        ; 父类
    extensions WSP RPAREN       ; 扩展
```

说明:

```
ruleids = ruleid / LPAREN WSP ruleidlist WSP RPAREN
ruleidlist = [ ruleid *( SP ruleid ) ]
ruleid = number
```

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

5.4.7 命名形式(name forms)

此概念在 LDAPv3 中没有提到,在 models 草案中被定义为"可以实现"的功能.目前还没有 LDAP server 支持此功能,在此仅作一般介绍,供用户了解 LDAP 协议的最新发展方向.

命名形式是"指明特定结构对象类的条目允许的 RDN. 1 个命名形式定义 1 个命名对象类和用来命名的 1 或多个属性类型. 命名形式是用来定义 DIT 结构规则的规范的原始片断".

每个命名形式指明要命名的结构对象类、需要的属性类型集合和允许的属性类型集合.1 个

属性类型不能在两个集合中同时出现.

5.4.7.1 格式

以下格式使用 ABNF 描述,符号含义参见本文的附录一"ABNF 符号说明".

```
NameFormDescription = LPAREN WSP
    numericoid                  ; OID
    [ SP "NAME" SP qdescrs ]    ; 名称
    [ SP "DESC" SP qdstring ]   ; 描述
    [ SP "OBSOLETE" ]           ; 废弃
    SP "OC" SP oid              ; 结构对象类
    SP "MUST" SP oids           ; 必须的属性类型
    [ SP "MAY" SP oids ]        ; 可选的属性类型
    extensions WSP RPAREN       ; 扩展
```

说明:

OBSOLETE 表示本属性类型已经被废弃,LDAP server 虽然保留它的定义,但并不使用.

第六章 LDIF

6.1 概要

此文档用于描述目录信息或修改目录信息的文件格式。LDIF（LDAP Data Interchange Format）即 LDAP 数据交换格式（源自 RFC2849 LDAP 数据交换格式——技术规范），是在 LDAP 目录服务器间导入、导出目录信息或描述应用于目录的一系列改变的文件。

6.2 背景及预期结果

这种公共交换格式描述了多种情况。例如，有人想把目录服务器中的内容导出到一个文件中，然后拿到其他的机器上导入另一台目录服务器。

再者，从先前的系统使用一种广为人知的交换格式的导入工具开发是非常便利的。用 awk 或 perl 就可以编写出一套相当简单的工具把人员信息数据库转换到 LDIF 文件。这个文件可以被导入到其他的目录服务器中，不管这个目录服务器使用的是那种内部数据库。

LDIF 格式最初是在密歇根大学 LDAP 实现中开发和使用的。最初，LDIF 用来描述目录的条目。后来，这种格式被扩展到表示目录条目的改变。

应用程序/目录的 MIME（多用途的网际邮件扩充协议）内容类型关联：

应用程序/目录的 MIME 内容类型是一种转换目录信息的通用的构架和格式，不依赖于任何一种具体的目录服务。LDIF 格式是一种更简单的格式，可以更易于创建，也可以被用来描述一组应用于目录的更改。

6.3 定义 LDAP 数据交换格式

LDIF 格式用于传送目录信息，或描述一组对目录条目的修改，LDIF 文件包括被行分隔符分割的一系列记录。一条记录包括一系列描述目录条目的行或一系列描述一组目录条目变化的行。一个 LDIF 文件指定一组目录条目,或一组应用于目录条目的更改，但二者不能兼顾。

修改目录条目的操作（添加、删除、修改及修改 RDN）和下面描述的修改记录的类型（"add","delete"," modify"及"modrdn"或"moddn"）是一一对应的，这一点是有意义的。并且目录服务器准许从 LDIF 更改记录到协议操作的直接翻译。

6.3.1 LDIF 的形式语法定义

以下定义使用 RFC2234 中指定的扩展 Backus-Naur。

```
LDIF-file                = LDIF-content / LDIF-changes
LDIF-content             = version-spec 1*(1*SEP LDIF-attrval-record)
LDIF-changes             = version-spec 1*(1*SEP LDIF-change-record)
LDIF-attrval-record      = dn-spec SEP 1*attrval-spec
LDIF-change-record       = dn-spec SEP *control changerecord
version-spec             = "version:" FILL version-number
version-number           = 1*DIGIT
                           ; 在此文档描述的 LDIF 格式中 version-number 必须是"1"
dn-spec                  = "dn:" (FILL distinguishedName /
                                  ":" FILL base64-distinguishedName)
distinguishedName        = SAFE-STRING
                           ; DN（distinguished name）——分辨名
base64-distinguishedName = BASE64-UTF8-STRING
                           ; 基于 base64 编码的 DN
rdn                      = SAFE-STRING
                           ; RDN（relative distinguished name）——相对分辨名
base64-rdn               = BASE64-UTF8-STRING
                           ; 基于 base64 编码的 RDN
control                  = "control:" FILL ldap-oid        ; control 类型
                           0*1(1*SPACE ("true" / "false")) ; 临界状态
                           0*1(value-spec)                 ; control 值
                           SEP
ldap-oid                 = 1*DIGIT 0*1("." 1*DIGIT)
                           ; LDAPOID
attrval-spec             = AttributeDescription value-spec SEP
value-spec               = ":" (    FILL 0*1(SAFE-STRING) /
                                ":" FILL (BASE64-STRING) /
                                "<" FILL url)
url                      = <a Uniform Resource Locator,
                           as defined in [6]>
AttributeDescription     = AttributeType [";" options]
AttributeType            = ldap-oid / (ALPHA *(attr-type-chars))
options                  = option / (option ";" options)
option                   = 1*opt-char
attr-type-chars          = ALPHA / DIGIT / "-"
opt-char                 = attr-type-chars
changerecord             = "changetype:" FILL
                           (change-add / change-delete /
                           change-modify / change-moddn)
change-add               = "add"                SEP 1*attrval-spec
change-delete            = "delete"             SEP
change-moddn             = ("modrdn" / "moddn") SEP
                           "newrdn:" (    FILL rdn /
                                      ":" FILL base64-rdn) SEP
                           "deleteoldrdn:" FILL ("0" / "1")  SEP
                           0*1("newsuperior:"
                           (    FILL distinguishedName /
                            ":" FILL base64-distinguishedName) SEP)
change-modify            = "modify"             SEP *mod-spec
```

```
mod-spec                = ("add:" / "delete:" / "replace:")
                          FILL AttributeDescription SEP
                          *attrval-spec
                          "-" SEP
SPACE                   = %x20
                          ; ASCII SP, space
FILL                    = *SPACE
SEP                     = (CR LF / LF)
CR                      = %x0D
                          ; ASCII 码 CR, 回车
LF                      = %x0A
                          ; ASCII 码 LF,换行
ALPHA                   = %x41-5A / %x61-7A
                          ; A-Z / a-z
DIGIT                   = %x30-39
                          ; 0-9
UTF8-1                  = %x80-BF
UTF8-2                  = %xC0-DF UTF8-1
UTF8-3                  = %xE0-EF 2UTF8-1
UTF8-4                  = %xF0-F7 3UTF8-1
UTF8-5                  = %xF8-FB 4UTF8-1
UTF8-6                  = %xFC-FD 5UTF8-1
SAFE-CHAR               = %x01-09 / %x0B-0C / %x0E-7F
                          ; 除 NUL, LF 及 CR 之外任何一个小于等于十进制 127 的
                          ; ASCII 码
SAFE-INIT-CHAR          = %x01-09 / %x0B-0C / %x0E-1F /
                          %x21-39 / %x3B / %x3D-7F
                          ; 除 NUL, LF, CR, SPACE, 冒号 (":", ASCII 码十进制 58)
                          ; 及小于号 ("<" , ASCII 码十进制 60) 之外任何一个小于
                          ; 等于十进制 127 的 ASCII 码
SAFE-STRING             = [SAFE-INIT-CHAR *SAFE-CHAR]
UTF8-CHAR               = SAFE-CHAR / UTF8-2 / UTF8-3 /
                          UTF8-4 / UTF8-5 / UTF8-6
UTF8-STRING             = *UTF8-CHAR
BASE64-UTF8-STRING      = BASE64-STRING
                          ; 必须被 base64 编码后的 UTF8-STRING
BASE64-CHAR             = %x2B / %x2F / %x30-39 / %x3D / %x41-5A /
                          %x61-7A
                          ; +, /, 0-9, =, A-Z, 及 a-z
BASE64-STRING           = [*(BASE64-CHAR)]
```

6.3.2 LDIF 语法的注意事项

1) 在本文档中描述的 LDIF 格式版本号必须为"1"。如果未标明版本号实现将以密歇根大学 ldap-3.3 支持的旧的 LDIF 文件格式解释内容。

2) 新起的一行的第一个字符不能为空。LDIF 文件中任何一个非空行包括注释行，可以通过插入一个行分割符(SEP)及一个 SPACE 进行折行。折行的第一个字符必须不被占用。任何一行如果以空开始就意味着它是上一行（非空行）的继续。多字节的 UTF-8 字符不应实现折行。

3) 任何以井号("#", ASCII 码 35)开始的行是注释行，解析 LDIF 文件时必须被忽略。

4) 任何 dn 或 rdn 所包含的字符不是由"SAFE-UTF8-CHAR"定义，或其起始字符不是由"SAFE-INIT-UTF8-CHAR"定义的必须以 base-64 编码。其它值可以用 base-64 编码。任何值所包含的字符不是由"SAFE-CHAR"定义，或其起始字符不是由"SAFE-INIT-CHAR"定义的必须以 base-64 编码。其它值可以用 base-64 编码。

5) 当 LDIF 文件直接包含 0 长度的属性值时，必须表示为 AttributeDescription ":" FILL SEP 的格式。例如，"seeAlso:"跟随着一个新行表示一个 0 长度的"seeAlso"属性值。除此之外，还允许引用 0 长度的 URL。

6) 当 URL 指向一个 attrval-spec 时，应使用下列惯例：
   a） 实现应支持 file:// URL 格式。参考文件的内容将被逐字逐句的导入 LDIF 文件
   b） 实现可以支持 file:// URL 格式。语法伴随的每一个支持的 URL 将归档到一个相关的适用说明中。
7) dn、rdn 和目录字符串的值语法必须为可用的 UTF-8 字符串。实现读取 LDIF 可以解释文件条目存储在其他的不同的字符集编码中，但不能产生非法的 UTF-8 的 LDIF 文件。

8) 以空格结尾的值或 dn 是 base-64 编码。

9) 如果 LDIF 文件中包括控制，实现可以有选择的忽略其中的一部分。这种情况可能需要 LDIF 文件中的修改描述为发送自 LDAPv2 的连接（LDAPv2 不支持控制），或来自远程服务器的不支持的特殊的控制。如果临界状态标为"true"则实现必须既包括控制不能连接到远程服务器。

10) 如果 attrval-spec，dn，rdn 是 base-64 编码，则编码规则必须使用以下异常：
   a） 基于 base64 输出流的请求必须以每行小于 76 个字符传送 LDIF 文件中的行可以只关联 2 中的折行规则。
   b） Base64 字符串可以包括 BASE64-CHAR 定义之外的字符，并被忽略。LDIF 拒绝任何除折行之外的无关字符。

6.3.3 LDAP 数据交换格式示例

例 1：两个条目的简单的 LDAP 文件

```
version: 1
dn: cn=Barbara Jensen, ou=Product Development, dc=airius, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Barbara Jensen
cn: Barbara J Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
telephonenumber: +1 408 555 1212
description: A big sailing fan.

dn: cn=Bjorn Jensen, ou=Accounting, dc=airius, dc=com
```

```
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Bjorn Jensen
sn: Jensen
telephonenumber: +1 408 555 1212
```

例 2：包括一个条目及一个折行的属性的文件

```
version: 1
dn:cn=Barbara Jensen, ou=Product Development, dc=airius, dc=com
objectclass:top
objectclass:person
objectclass:organizationalPerson
cn:Barbara Jensen
cn:Barbara J Jensen
cn:Babs Jensen
sn:Jensen
uid:bjensen
telephonenumber:+1 408 555 1212
description:Babs is a big sailing fan, and travels extensively in sea
 rch of perfect sailing conditions.
title:Product Manager, Rod and Reel Division
```

例 3：值包括 base-64-编码的文件

```
version: 1
dn: cn=Gern Jensen, ou=Product Testing, dc=airius, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Gern Jensen
cn: Gern O Jensen
sn: Jensen
uid: gernj
telephonenumber: +1 408 555 1212
description:: V2hhdCBhIGNhcmVmdWwgcmVhZGVyIHlvdSBhcmUhICBUaGlzIHZhbHVl
IGlzIGJhc2UtNjQtZW5jb2RlZCBiZWNhdXNlIGlOIGhhcyBhIGNvbnRyb2wgY2hhcmFjdG
VyIGluIGlOIChhIENSKS4NICBCeSBOaGUgd2F5LCB5b3Ugc2hvdWxkIHJlYWxseSBnZXQg
b3VOIG1vcmUu
```

例 4：一个已有 UTF-8-编码的属性值，包括语言标签的文件。注释指明了 UTF-8-编码属性内容及 dn。

```
version: 1
dn:: b3U95Za25qWt6YOoLG89QWlyaXVz
# dn:: ou=<JapaneseOU>,o=Airius
objectclass: top
objectclass: organizationalUnit
ou:: 5Za25qWt6YOo
# ou:: <JapaneseOU>
ou;lang-ja:: 5Za25qWt6YOo
# ou;lang-ja:: <JapaneseOU>
```

```
ou;lang-ja;phonetic:: 44GI44GE44GO44KH44GG44G2

# ou;lang-ja:: <JapaneseOU_in_phonetic_representation>
ou;lang-en: Sales
description: Japanese office

dn:: dWlkPXJvZ2FzYXdhcmEsb3U95Za25qWt6YOoLG89QWlyaXVz
# dn:: uid=<uid>,ou=<JapaneseOU>,o=Airius
userpassword: {SHA}O3HSv1MusyL4kTjP+HKI5uxuNoM=
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
uid: rogasawara
mail: rogasawara@airius.co.jp
givenname;lang-ja:: 44Ot44OJ44OL44O8
# givenname;lang-ja:: <JapaneseGivenname>
sn;lang-ja:: 5bCP56yg5Y6f
# sn;lang-ja:: <JapaneseSn>
cn;lang-ja:: 5bCP56yg5Y6fIOODreODieODi+ODvA==
# cn;lang-ja:: <JapaneseCn>
title;lang-ja:: 5Za25qWt6YOoIOmDqOmVtw==
# title;lang-ja:: <JapaneseTitle>
preferredlanguage: ja
givenname:: 44Ot44OJ44OL44O8
# givenname:: <JapaneseGivenname>
sn:: 5bCP56yg5Y6f
# sn:: <JapaneseSn>
cn:: 5bCP56yg5Y6fIOODreODieODi+ODvA==
# cn:: <JapaneseCn>
title:: 5Za25qWt6YOoIOmDqOmVtw==
# title:: <JapaneseTitle>
givenname;lang-ja;phonetic:: 44KN44Gp44Gr44O8
# givenname;lang-ja;phonetic::
<JapaneseGivenname_in_phonetic_representation_kana>
sn;lang-ja;phonetic:: 44GK44GM44GV44KP44KJ
# sn;lang-ja;phonetic:: <JapaneseSn_in_phonetic_representation_kana>
cn;lang-ja;phonetic:: 44GK44GM44GV44KP44KJIOOCjeOBqeOBq+ODvA==
# cn;lang-ja;phonetic:: <JapaneseCn_in_phonetic_representation_kana>
title;lang-ja;phonetic:: 44GI44GE44GO44KH44GG44G2IOOBtuOBoeOCh+OBhg==
# title;lang-ja;phonetic::
# <JapaneseTitle_in_phonetic_representation_kana>
givenname;lang-en: Rodney
sn;lang-en: Ogasawara
cn;lang-en: Rodney Ogasawara
title;lang-en: Sales, Director
```

例 5：引用外部文件

```
version: 1
dn: cn=Horatio Jensen, ou=Product Testing, dc=airius, dc=com
```

```
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Horatio Jensen

cn: Horatio N Jensen
sn: Jensen
uid: hjensen
telephonenumber: +1 408 555 1212
jpegphoto:< file:///usr/local/directory/photos/hjensen.jpg
```

例 6：具有修改记录及注释的文件

```
version: 1
# Add a new entry
dn: cn=Fiona Jensen, ou=Marketing, dc=airius, dc=com
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Fiona Jensen
sn: Jensen
uid: fiona
telephonenumber: +1 408 555 1212
jpegphoto:< file:///usr/local/directory/photos/fiona.jpg

# Delete an existing entry
dn: cn=Robert Jensen, ou=Marketing, dc=airius, dc=com
changetype: delete

# Modify an entry's relative distinguished name
dn: cn=Paul Jensen, ou=Product Development, dc=airius, dc=com
changetype: modrdn
newrdn: cn=Paula Jensen
deleteoldrdn: 1

# Rename an entry and move all of its children to a new location in
# the directory tree (only implemented by LDAPv3 servers).
dn: ou=PD Accountants, ou=Product Development, dc=airius, dc=com
changetype: modrdn
newrdn: ou=Product Development Accountants
deleteoldrdn: 0
newsuperior: ou=Accounting, dc=airius, dc=com

# Modify an entry: add an additional value to the postaladdress
# attribute, completely delete the description attribute, replace
# the telephonenumber attribute with two values, and delete a specific
# value from the facsimiletelephonenumber attribute
dn: cn=Paula Jensen, ou=Product Development, dc=airius, dc=com
changetype: modify
add: postaladdress
```

```
postaladdress: 123 Anystreet $ Sunnyvale, CA $ 94086
-

delete: description
-
replace: telephonenumber
telephonenumber: +1 408 555 1234
telephonenumber: +1 408 555 5678
-
delete: facsimiletelephonenumber
facsimiletelephonenumber: +1 408 555 9876
-

# Modify an entry: replace the postaladdress attribute with an empty
# set of values (which will cause the attribute to be removed), and
# delete the entire description attribute. Note that the first will
# always succeed, while the second will only succeed if at least
# one value for the description attribute is present.
dn: cn=Ingrid Jensen, ou=Product Support, dc=airius, dc=com
changetype: modify
replace: postaladdress
-
delete: description
-
```

例 7：带有控制的 LDIF 文件

```
version: 1
# Delete an entry. The operation will attach the LDAPv3
# Tree Delete Control defined in [9]. The criticality
# field is "true" and the controlValue field is
# absent, as required by [9].
dn: ou=Product Development, dc=airius, dc=com
control: 1.2.840.113556.1.4.805 true
changetype: delete
```

6.4 安全考虑

对于典型的目录应用程序，LDIF 文件通常会涉及到一些敏感的私人数据。通过适当的方法处理可以保护 LDIF 文件中的隐私数据。

":<"可以指向一个外部文件，以供处理 LDIF 文件时使用，因此在接受外部文件时要相当谨慎。一个“特洛伊木马”LDIF 文件可以以敏感内容命名，然后诱使接收者将其条目导入目录，再利用非法条目通过 LDAP 读取目录。

对于 LDIF 文件 LDIF 不提供任何方法加载认证信息。LDIF 文件的使用者必须仔细验证从外部接受到的 LDIF 文件的完整性。

第七章 LDAP 查询过滤

7.1 概述

轻量级目录访问协议（Lightweight Directory Access Protocol——LDAP）定义了对 LDAP

服务器进行查询过滤的网络描述。一些应用程序可能使用某种通用的方法把查询过滤表示为人可识别的结构。本文档以人可识别的字符串表示过滤条件。

本文档扩展了 LDAP 过滤字符串定义包括支持 LDAPv3 扩展匹配过滤及表示 LDAP 查询过滤可能的全部范围。

7.2 LDAP 查询过滤定义

```
Filter ::= CHOICE {
        and                             [0] SET OF Filter,
        or                              [1] SET OF Filter,
        not                             [2] Filter,
        equalityMatch           [3] AttributeValueAssertion,
        substrings              [4] SubstringFilter,
        greaterOrEqual  [5] AttributeValueAssertion,
        lessOrEqual             [6] AttributeValueAssertion,
        present                 [7] AttributeDescription,
        approxMatch             [8] AttributeValueAssertion,
        extensibleMatch[9] MatchingRuleAssertion
}

SubstringFilter ::= SEQUENCE {
        type            AttributeDescription,
        SEQUENCE OF CHOICE {
                initial         [0] LDAPString,
                any             [1] LDAPString,
                final           [2] LDAPString
        }
}

AttributeValueAssertion ::= SEQUENCE {
        attributeDesc           AttributeDescription,
        attributeValue          AttributeValue
}

MatchingRuleAssertion ::= SEQUENCE {
        matchingRule            [1] MatchingRuleID OPTIONAL,
        type                            [2] AttributeDescription OPTIONAL,
        matchValue              [3] AssertionValue,
        dnAttributes            [4] BOOLEAN DEFAULT FALSE
}

AttributeDescription ::= LDAPString

AttributeValue ::= OCTET STRING

MatchingRuleID ::= LDAPString

AssertionValue ::= OCTET STRING

LDAPString ::= OCTET STRING
```

LDAPString 应为 ISO10646 字符集的 UTF-8 编码。AttributeDescription 是描述在 RFC2251 中定义的属性的字符串。AttributeValue 与 AssertionValue 在 RFC2252 中定义。过滤器要经过 BER（基本编码规则）编码后在网络上传输。

7.3 字符串查询过滤定义

LDAP 查询过滤字符串被以下语法定义。过滤器格式使用前缀符号。

```
filter        = "(" filtercomp ")"
filtercomp    = and / or / not / item
and           = "&" filterlist
or            = "|" filterlist
not           = "!" filter
filterlist    = 1*filter
item          = simple / present / substring / extensible
simple        = attr filtertype value
filtertype    = equal / approx / greater / less
equal         = "="
approx        = "~="
greater       = ">="
less          = "<="
extensible    = attr [":dn"] [":" matchingrule] ":=" value
              / [":dn"] ":" matchingrule ":=" value
present       = attr "=*"
substring     = attr "=" [initial] any [final]
initial             = value
any           = "*" *(value "*")
final         = value
attr          = AttributeDescription
matchingrule = MatchingRuleId
value         = AttributeValue
```

如果值包含下列字符

Character	ASCII value
*	0x2a
(	0x28
)	0x29
\	0x5c
NUL	0x00

这些字符必须以反斜杠'\'(ASCII 0x5c)加上由两位 16 进制数表示的 ASCII 码编码。

这一简单的转义机制除去过滤器解析算法和允许任何一个在 LDAP 中表示的过滤器表示为'\0'结尾的字符串。除了上面列出的字符之外其他的字符可以在这一机制中使用转义符，如不可打印字符。

例如，要求检查所有符合"cn"属性值包含字符"*"的过滤条件应表示为"(cn=*\2a*)"。

注意，以上所述语法中的子串和当前产品可以产生"attr=*"结构，这种结构仅仅用来指示当前过滤器。

7.4 示例

此段给出几个查询过滤的例子。

```
(cn=Babs Jensen)
(!(cn=Tim Howes))
(&(objectClass=Person)(|(sn=Jensen)(cn=Babs J*)))
(o=univ*of*mich*)
```

下面几个例子举例说明扩展匹配:

```
(cn:1.2.3.4.5:=Fred Flintstone)
(sn:dn:2.4.6.8.10:=Barney Rubble)
(o:dn:=Ace Industry)
(:dn:2.4.6.8.10:=Dino)
```

上面第二个例子指明在使用符号":dn"时进行比较指定匹配规则"2.4.6.8.10"条目 dn 的属性应该考虑作为当前评估匹配的一部分。

第三个例子指明一个相等匹配，在做匹配时除了 DN 组成应被考虑成条目的一部分。

第四个例子是一个应该被任何一个支持给定匹配规则属性应用的（从 attr 以左）。属性所支持的 DN 中包含的匹配规则也应该被考虑。

下面的示例说明转义符机制的使用:

```
(o=Parens R Us \28for all your parenthetical needs\29)
(cn=*\2A*)
(filename=C:\5cMyFile)
(bin=\00\00\00\04)
(sn=Lu\c4\8di\c4\87)
```

第一个例子展示了如何用转义符机制表示圆括号。第二个例子显示如何在一个值中显示"*"，以防被解释为通配符。第三个指明的反斜杠转义符。

第四个例子显示的是有四位字节的查询过滤条件值为 0x00000004，指明转义机制表示包括空字符在内的任何数据的用法。

最后一个例子阐明转义机制表示非 ASCII 码的 UTF-8 字符的用法。

7.5 安全考虑

这个备忘录描述 LDAP 查询过滤表示的字符串。当描述本身含有未知的安全含义时，LDAP 查询过滤器执行，这将被 LDAP 服务器认为选择需要返回的条目的数据。LDAP 服务器应该通过维护未经认证的访问来仔细保护数据。