OpenLDAP-Tutorial I: Konfiguration und Betrieb des freien Verzeichnisdienstes

OpenLDAP-Tutorial I: Konfiguration und Betrieb des freien Verzeichnisdienstes

[expand title="mehr lesen…"]

OpenLDAP-Tutorial I: Konfiguration und Betrieb des freien Verzeichnisdienstes

Schnell nachgeschlagen

Mark Pröhl

Ein sinnvolles Identity Management lässt sich in größeren Umgebungen nicht ohne Verzeichnisdienst realisieren. Neben Microsofts Active Directory ist das freie OpenLDAP einer der bekanntesten Vertreter dieser Gattung. Wie es sich einrichten lässt, zeigt dieser Auftakt eines dreiteiligen Tutorials.

iX-TRACT

OpenLDAP ist eine ernst zu nehmende Open-Source-Alternative zu kommerziellen Verzeichnisdiensten.

LDAP-Verzeichnisse speichern Daten in hierarchisch angeordneten Objekten und Attributen. Das Schema eines Verzeichnisdienstes legt fest, welche Klassen von Objekten und welche Attribute es gibt.

Mit einer einfachen Konfigurationsdatei ist der Server-Daemon *slapd* schnell betriebsbereit.

Zum Durchsuchen und Verwalten des Datenbestands nutzen die OpenLDAP-Kommandozeilenwerkzeuge das LDAP Data Interchange Format (LDIF).

Verzeichnisdienste auf Basis des Lightweight Directory Access Protocol (LDAP v3) spielen eine zentrale Rolle im Identity Management, wo man sie als Quelle für Benutzeridentitäten, Authentifizierungs- und Autorisierungsdaten einsetzt. Prominentes Beispiel für einen weitverbreiteten Verzeichnisdienst mit diesem Fokus ist Active Directory. Daneben existieren etliche andere kommerzielle sowie freie LDAP-Implementierungen, die häufig universeller einsetzbar sind. Eine davon – OpenLDAP – ist das Thema dieses *iX*-Tutorials.

OpenLDAP ist eines der renommiertesten Open-Source-Projekte und bietet eine ausgereifte und standardkonforme LDAP-Referenzimplementierung. Der Verzeichnisdienst ist Bestandteil aller gängigen Linux-Distributionen und daher sehr weit verbreitet. OpenLDAP spielt durchaus in derselben Liga wie die zahlreichen kommerziellen Implementierungen diverser Hersteller: Die Software bietet gängige Funktionen wie Ausfallsicherheit oder Multimaster-Betrieb und übertrifft manche kommerzielle Konkurrenz in Sachen Performance und Skalierbarkeit (siehe „Onlinequellen“, [a]).

Im ersten Teil geht es um die Grundlagen von LDAP, konkret am Beispiel von OpenLDAP. Das Hauptaugenmerk liegt dabei auf dem Datenmodell, die Konfiguration des Dienstes selbst erfolgt zunächst nur rudimentär. Die folgenden Teile widmen sich dann

im Detail der dynamischen Konfiguration, den Sicherheitsaspekten und den Themen Replikation und Synchronisation.

Derzeit liegt OpenLDAP in der Version 2.4.42 vor, Debian, Red Hat/CentOS und die meisten anderen Linux-Distributionen setzen noch ältere Releases ein. Aktuelle Pakete bekommt man distributionsübergreifend vom LDAP Tool Box Project (LTB, [b]). Das *iX*-Tutorial verwendet diese LTB-Pakete, wobei die Beispiele unter CentOS 7 mit *openldap-ltb-2.4.42-1* liefen.

Grundinstallation mit LTB

Listing 1: /etc/openldap/slapd.conf

```
include   /usr/local/openldap/etc/openldap/schema/core.schema
include
/usr/local/openldap/etc/openldap/schema/cosine.schema
include
/usr/local/openldap/etc/openldap/schema/inetorgperson.schema
pidfile   /usr/local/openldap/var/run/slapd.pid
argsfile  /usr/local/openldap/var/run/slapd.args
database  mdb
suffix    "o=tutorial"
directory /var/lib/ldap/tutorial
rootdn    "cn=root,o=tutorial"
rootpw    "P@ssw0rd"
access    to * by * read
```

Los geht es mit der Hauptkomponente des OpenLDAP-Verzeichnisdienstes, dem Daemon *slapd*. Dessen zeitgemäße Konfiguration und auch deren Sicherheitsaspekte sind Thema der nächsten Tutorialteile, an dieser Stelle belässt der Artikel es bei der sehr einfach gehaltenen *slapd.conf* aus Listing 1. Die verwendet als Namensraum für die LDAP-Daten *o=tutorial*, die *access*-Regel erlaubt anonymes Lesen für jedermann, die Definition eines *rootdn* und dessen Passwort gestattet diesem beliebige Schreibzugriffe. Gerade am Klartextpasswort erkennt man, dass diese Konfiguration so noch völlig unsicher ist und

in den folgenden Tutorialteilen überarbeitet werden muss.

Installation und Konfiguration

Listing 3: /etc/yum.repos.d/ltb-project.repo

```
[ltb-project]
name=LTB project packages
baseurl=http://ltb-project.org/rpm/$releasever/$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-LTB-project
```

Mit folgenden Schritten lässt sich unter CentOS OpenLDAP aus dem LTB-Projekt einrichten. Zuerst erweitert man die Yum-Konfiguration um das LTB-Repository, indem man die in Listing 3 gezeigte Datei */etc/yum.repos.d/ltb-project.repo* anlegt und anschließend die folgenden Kommandos ausführt:

```
# yum update
# rpm --import http://ltb-project.org/wiki/
 lib/RPM-GPG-KEY-LTB-project
```

Dann installiert *yum install openldap-ltb* die benötigten Pakete. Danach ist */etc/default/slapd* wie in Listing 2 zu modifizieren und *slapd.conf* gemäß Listing 1 anzulegen. Jetzt legt man das Verzeichnis an, in dem die Daten für das Tutorial liegen sollen:

```
mkdir -p /var/lib/ldap/tutorial
chown ldap:ldap /var/lib/ldap/tutorial
chmod 0700 /var/lib/ldap/tutorial
```

Die beiden Aufrufe

```
systemctl start slapd.service
chkconfig slapd on
```

starten und aktivieren den Dienst.

Listing 2: /etc/default/slapd

```
IP="127.0.0.1"
```

```
PORT="389"
SLAPD_PATH="/usr/local/openldap"
SLAPD_PID_FILE="$SLAPD_PATH/var/run/slapd.pid"
SLAPD_CONF="/etc/openldap/slapd.conf"
SLAPD_CONF_DIR=""
SLAPD_SERVICES="ldap://$IP:$PORT"
SLAPD_PARAMS=""
SLAPD_BIN="$SLAPD_PATH/libexec/slapd"
SLAPD_USER="ldap"
SLAPD_GROUP="ldap"
```

Der Kasten „Installation und Konfiguration“ beschreibt die Schritte für das Einrichten des Dienstes mit LTB. Unter CentOS befindet sich *slapd.conf* im Verzeichnis */etc/openldap*. LTB konfiguriert diesen Pfad über den Parameter *SLAPD_CONF* in */etc/default/slapd* (Listing 2). Als Ergebnis läuft der *slapd* mit folgenden Parametern:

```
slapd -h ldap://127.0.0.1:389 -f
 /etc/openldap/slapd.conf -u ldap -g ldap
```

Der *slapd*-Parameter *–f* definiert den Pfad zur Konfigurationsdatei; *–u* und *–g* definieren den lokalen Benutzer-Account und dessen Gruppe, unter dem *slapd* laufen soll. Diese lauten beide *ldap*. *–h* legt fest, wie – also über welches Netzwerkinterface und welchen Port – sich der Daemon ansprechen lässt. Die korrespondierenden Parameter *IP=“127.0.0.1˝* sowie *PORT=“389˝* in Listing 2 führen dazu, dass *slapd* zunächst nur am Loopback-Interface auf den Standard-LDAP-Port 389 hört.

Erste Tests kann man mit dem Kommando *ldapsearch* anstoßen:

```
ldapsearch -H ldap://127.0.0.1
 -x -LLL -b "" -s base +
```

Listing 4: Erste LDAP-Suche gegen den RootDSE

```
# ldapsearch -H ldap://127.0.0.1 -x -LLL -b "" -s base +
dn:
structuralObjectClass: OpenLDAProotDSE
```

```
configContext: cn=config
namingContexts: o=tutorial
supportedControl: 1.3.6.1.4.1.4203.1.9.1.1
supportedControl: 2.16.840.1.113730.3.4.18
supportedControl: 2.16.840.1.113730.3.4.2
supportedControl: 1.3.6.1.4.1.4203.1.10.1
supportedControl: 1.3.6.1.1.22
supportedControl: 1.2.840.113556.1.4.319
supportedControl: 1.2.826.0.1.3344810.2.3
supportedControl: 1.3.6.1.1.13.2
supportedControl: 1.3.6.1.1.13.1
supportedControl: 1.3.6.1.1.12
supportedExtension: 1.3.6.1.4.1.4203.1.11.1
supportedExtension: 1.3.6.1.4.1.4203.1.11.3
supportedExtension: 1.3.6.1.1.8
supportedFeatures: 1.3.6.1.1.14
supportedFeatures: 1.3.6.1.4.1.4203.1.5.1
supportedFeatures: 1.3.6.1.4.1.4203.1.5.2
supportedFeatures: 1.3.6.1.4.1.4203.1.5.3
supportedFeatures: 1.3.6.1.4.1.4203.1.5.4
supportedFeatures: 1.3.6.1.4.1.4203.1.5.5
supportedLDAPVersion: 3
entryDN:
subschemaSubentry: cn=Subschema
```

liefert als Ergebnis den sogenannten RootDSE (Listing 4). Dieser Begriff steht für den DSA-specific Entry, einen Eintrag, in dem LDAP-Server (auch Directory System Agent oder kurz DSA genannt) Informationen für Clients vorhalten. Neben den LDAP-Features, -Controls und -Extensions, die jeweils mit einem Object Identifier (OID) aufgeführt werden, enthält der RootDSE auch die Information, für welche LDAP-Namensräume dieser Server zuständig ist: *namingContexts: o=tutorial*. Auch wo man das Schema abfragen kann, erfährt man hier: *subschemaSubentry: cn=Subschema*.

Eine weitere LDAP-Suche nach den Daten unter *o=tutorial* kann nun mit dem Kommando *ldapsearch –H ldap://127.0.0.1 –x –b o=tutorial* erfolgen. Das Ergebnis wird dabei zunächst *No such object* lauten – denn es existieren zu diesem Zeitpunkt noch

keine Nutzdaten im Tutorialverzeichnis.

Daten im LDAP

Läuft der Dienst nach dem Abarbeiten der Installationsschritte gemäß Kasten „Installation und Konfiguration“, sollte man sich als Nächstes mit den dort zu speichernden Daten beschäftigen. Es geht also um die Frage, welche Daten man „im LDAP“ speichern kann und in welcher Form dies geschehen soll. Dazu zunächst etwas Theorie zum Datenmodell, das RFC 4512 [c] im Detail beschreibt.

LDAP-Verzeichnisse speichern ihre Daten in Form von Objekten oder auch Einträgen. Jedes Objekt besteht aus Attributen, denen man Werte zuordnen kann. Objekte gehören sogenannten Objektklassen an. Die legen fest, welche Attribute das Objekt haben kann (MAY-Attribute) und welche es haben muss (MUST-Attribute). Beispielsweise muss nicht jedes Personenobjekt im LDAP eine Telefonnummer haben, aber Personen ohne Namen sind nicht erlaubt.

Welche Informationen sich in einem Verzeichnisdienst befinden, legt dessen Schema fest. Das definiert genau, welche Attribute und Objektklassen zur Verfügung stehen. Des Weiteren definiert es auch Syntaxregeln für die Attributwerte und Vergleichsoperationen, die bei Suchen im Verzeichnis Verwendung finden.

Grundsätzlich lassen sich in einem LDAP-Verzeichnis beliebige Daten speichern. Man kann also beliebige Dinge durch LDAP-Objekte darstellen und ihre Eigenschaften in passenden Attributen speichern. Ein weitverbreiteter Anwendungsfall von Verzeichnisdiensten besteht aber im Speichern von Daten zu Personen, Accounts und deren Berechtigungen; Letzteres in Form von Rollen oder Gruppen. Die Beispiele im Tutorial konzentrieren sich daher auf dieses Anwendungsszenario.

Namen und Hierarchien

Zum Referenzieren eines Objekts benötigt dieses einen Namen. Dazu ist zunächst der Begriff Relative Distinguished Name (RDN) von Bedeutung. LDAP sieht vor, den RDN eines Objekts aus dessen Attribut-Wert-Paaren zu konstruieren. Der RDN hat dann die Form *attribut=wert*. Üblicherweise bestimmt genau ein Attribut-Wert-Paar den RDN, es ist aber auch möglich, den RDN aus mehreren Paaren zusammenzusetzen. So können RDNs der folgenden Form entstehen: *attribut1=wert1+attribut2=wert2*.

Nun sind LDAP-Objekte hierarchisch in einer Baumstruktur organisiert. So hat jedes Objekt ein Elternobjekt (mit Ausnahme der Wurzel) und kann Kindobjekte haben. Dadurch entsteht der Directory Information Tree, kurz (DIT).

Der RDN alleine referenziert ein Objekt daher nur eindeutig unterhalb eines Elternobjekts. Für ein eindeutiges Referenzieren eines Objekts im DIT fasst man seinen RDN und die Liste der RDNs seiner Eltern, Großeltern und aller weiteren Vorfahren bis zur Wurzel des DIT zusammen. So entsteht der Distinguished Name oder kurz DN. Der bezeichnet das Objekt eindeutig im gesamten DIT. Ein Objekt für das Tutorial könnte also den folgenden DN haben: *attribut=wert,attribut=wert,o=tutorial*.

Im Detail entsteht diese DIT-Struktur wie folgt: Der Name der Wurzel soll *o=tutorial* lauten. Dabei ist *o* (kurz für *organization*) der Name des Attributs, *tutorial* sein Wert und *o=tutorial* das namengebende Attribut-Wert-Paar, also der RDN der Wurzel.

In der Praxis kann man den DIT an die Hierarchie der betreffenden Organisation anlehnen und wie hier den Namen des Wurzelobjekts dem Organisationsnamen entnehmen. Dabei ist aber die Eindeutigkeit dieses Namens nicht garantiert. Ein anderer Weg, das Wurzelobjekt zu benennen, ist ebenfalls sehr üblich: Man lehnt es einfach an den DNS-Namen an. LDAP sieht dafür das

Attribut *dc* (kurz für *domainComponent*) vor. Lautet der DNS-Name *tutorial.org*, wäre *dc=tutorial,dc=org* ebenfalls ein passender Name für die Wurzel des DIT dieses Tutorials.

Auch unterhalb der Wurzel lässt sich der LDAP-Baum nach verschiedenen Gesichtspunkten strukturieren: beispielsweise kann man organisatorische oder funktionale Aspekte heranziehen. Für die Verzweigungen verwendet man in der Praxis gerne Objekte der Klasse *organizationalUnit* und benennt diese mit dem *ou*-Attribut. Für organisatorische Einheiten einer Firma, wie Vertrieb oder Personalabteilung, wären also Zweige mit den RDNs *ou=sales* oder *ou=hr* durchaus üblich. Beim alternativen Strukturieren nach funktionalen Gesichtspunkten verwendet man ebenfalls *ou*. So ist es üblich, Mitarbeitern Benutzerobjekte unterhalb eines Zweigs mit dem Namen *ou=people* zu geben und Gruppen in *ou=groups* abzulegen.

Bei einer Mischung von organisatorischer und funktionaler Strukturierung erhält man Teilbäume wie *ou=people,ou=marketing,o=tutorial*. So entsteht schnell eine tiefere Baumstruktur, die der Übersichtlichkeit durchaus dienen kann. In der Praxis hat es sich aber als sinnvoll erwiesen, nicht unnötig zu verzweigen, sondern den DIT möglichst flach zu gestalten. Gibt es also keinen zwingenden Grund für eine tiefere Verzweigung, reichen *ou=people,o=tutorial* und *ou=groups,o=tutorial* völlig aus.

DIT-Struktur für das Tutorial

Listing 5: Die Struktur des Tutorial-DIT

```
dn: o=tutorial
changetype: add
objectclass: top
objectclass: organization
o: tutorial

dn: ou=people,o=tutorial
changetype: add
```

```
objectclass: top
objectclass: organizationalUnit
ou: people

dn: ou=groups,o=tutorial
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: groups

dn: ou=admins,o=tutorial
changetype: add
objectclass: top
objectclass: organizationalUnit
ou: admins
```

Neben *ou=people* für Personenobjekte wird das Tutorial auch noch Gruppen und LDAP-Administratoren unterhalb von *ou=groups* und *ou=admins* verwalten. Listing 5 beschreibt diese einfache Struktur. Neben den drei *ou*-Objekten ist dort auch das Wurzelobjekt *o=tutorial* selbst enthalten.

Das Listing verwendet zum Beschreiben das sogenannte LDAP Data Interchange Format (LDIF, [d]), ein einfaches 7-Bit-ASCII-Format, mit dem sich LDAP-Objekte und Operationen an ihnen beschreiben lassen. LDIF-Dateien enthalten durch Leerzeilen getrennte Textblöcke, die jeweils ein Objekt oder Operationen daran beschreiben. Die erste Zeile pro Block enthält den DN, die zweite kann die Operation (*changetype*) enthalten. Erlaubte Werte sind hier beispielsweise *add*, *delete* oder *modify*.

Speichert man diese Beschreibung in einer Datei mit dem Namen *listing-5.ldif*, lässt sich der Inhalt mit folgendem Kommando dem DIT hinzufügen:

```
ldapadd -x -H ldap://127.0.0.1 -D cn=root,
 o=tutorial -w P@ssw0rd -f listing-5.ldif
```

Damit steht die Struktur des DIT fest.

Weitere Nutzdaten in den Verzeichnisbaum integrieren

Listing 6: Zwei Personenobjekte

```
dn: cn=Max Mustermann,ou=people,o=tutorial
changetype: add
objectclass: top
objectclass: person
cn: Max Mustermann
sn: Mustermann

dn: cn=Erika Musterfrau,ou=people,o=tutorial
changetype: add
objectclass: top
objectclass: person
cn: Erika Musterfrau
sn: Musterfrau
```

Traditionell dienen LDAP-Verzeichnisse zum Erfassen von Personendaten. Dazu gehören Attribute wie Vor- und Nachname, Geburtstag, aber auch Anschrift und Telefonnummer, E-Mail-Adresse et cetera. Listing 6 enthält zwei Personenobjekte. Die Beispiele sind bewusst einfach gehalten und sollen im Folgenden noch erweitert werden. Beide Personen gehören der Objektklasse *person* an. Für solche Objekte sieht LDAP hauptsächlich Namensattribute vor: Nachname (*sn*, kurz für *surname*) und *cn* (*commonName*). Darüber hinaus könnten sie über ein Passwort (*userPassword*) oder eine Telefonnummer (*telephoneNumber*) verfügen, aber nur *sn* und *cn* sind zwingend erforderlich. Das vorliegende Beispiel verwendet jeweils *cn* als namengebendes Attribut im RDN; *sn* wäre ebenfalls möglich, wenn auch unüblich.

Diese Beschreibung speichert man in einer Datei mit dem Namen *listing-6.ldif* und fügt sie wie zuvor per *ldapadd* hinzu. Damit läuft der Verzeichnisdienst und bietet bereits eine einfache Baumstruktur mit zwei Personenobjekten an. Das genügt schon, um sich ein wenig mehr mit den Suchmöglichkeiten zu

beschäftigen, die LDAP bietet.

Anatomie einer LDAP-Suche

Listing 7: Eine rekursive LDAP-Suche

```
# ldapsearch -H ldap://127.0.0.1 -b o=tutorial -s sub -x -LLL
'(objectClass=*)' cn

 dn: o=tutorial

 dn: ou=people,o=tutorial

 dn: cn=Max Mustermann,ou=people,o=tutorial
 cn: Max Mustermann

 dn: cn=Erika Musterfrau,ou=people,o=tutorial
 cn: Erika Musterfrau

 dn: ou=groups,o=tutorial

 dn: ou=admins,o=tutorial
```

Listing 7 stellt eine typische LDAP-Suchoperation mit dem OpenLDAP-Werkzeug *ldapsearch* vor: *–H* gibt die LDAP-URI, also im Wesentlichen den Hostnamen oder die IP-Adresse des Servers an, *–b* die Suchbasis. Die Beispielsuche erfolgt also ab dem Wurzelobjekt *o=tutorial* – man könnte die Suche über *–b* auch auf einen Teilbaum einschränken. Der Parameter *–s* gibt den Scope der Suchoperation vor, mögliche Werte sind *sub* (durchsucht den DIT rekursiv unterhalb der angegebenen Suchbasis), *one* (findet nur direkte Kindobjekte der Suchbasis) und *base* (liefert nur das Basisobjekt selbst als Suchergebnis).

–x dient der Authentifizierung per sogenanntem Simple Bind, bei dem der LDAP-Client einen Namen (den Bind-DN) und das zugehörige Passwort benötigt. Mit *–D* könnte man den Bind-DN angeben, mit *–w* dessen Passwort. Anstelle von *–w* akzeptieren die OpenLDAP-Werkzeuge das Passwort auch interaktiv (über *–W*)

oder aus einer Passwortdatei (mit *–y*). Ohne diese zusätzlichen Parameter führt *–x* zu einem anonymen Simple Bind. Der Parameter *–LLL* ist reine Kosmetik: Er reduziert die Menge der ausgegebenen LDIF-Daten etwas.

Des Weiteren enthält die Kommandozeile aus Listing 7 einen Suchfilter und eine Liste der gesuchten Attribute. *‚(objectClass=*)‘* ist der einfachste denkbare Suchfilter, der auf alle Objekte im DIT passt. Weil *cn* das einzige Suchattribut auf der Kommandozeile ist, ist auch nur dieses im Suchergebnis enthalten. Der Bind-DN *cn=root,o=tutorial* kommt in der Ausgabe nicht vor, denn den gibt es nicht im DIT, sondern lediglich als Konfigurationsparameter.

So liefert *ldapsearch* einen Überblick über die Objekte und auch ihre Anordnung im DIT. Für den Einstieg in LDAP empfiehlt sich die Kommandozeile. Sie soll hier im Tutorial ausschließlich zum Einsatz kommen. Möchte man im täglichen Betrieb mehr Komfort, greift man zu einem der grafischen LDAP-Browser, wie Apache Directory Studio, Softeras LDAP Administrator, dem Userbooster von maxcrc oder dem webbasierten Web2LDAP.

Weitere LDAP-Operationen

Beide Personenobjekte besitzen bislang nur die nötigsten Attribute. Es wäre sicherlich nützlich, wenn man ihre Kontaktinformationen und Adressdaten ebenfalls speichern könnte. Sollen mit diesen Personenobjekten auch Betriebssystem-Accounts, beispielsweise für Linux, verbunden sein, kämen sogenannte POSIX-Attribute dazu. LDAP bietet nicht nur die Möglichkeit, zu suchen oder ganze Objekte mit der Add-Operation (über *ldapadd*) hinzuzufügen. Es lassen sich auch Objekte anpassen. Die Modify-Operation erlaubt das Hinzufügen, Ändern und Entfernen einzelner Attribute eines Objekts. Beschreibt man solche Modifikationen in einer LDIF-Datei, kann man diese unter OpenLDAP mit dem Werkzeug *ldapmodify* anwenden.

Listing 8: LDIF-Datei für LDAP-Modify-Operation

```
dn: cn=Max Mustermann,ou=people,o=tutorial
changetype: modify
add: telephoneNumber
telephoneNumber: +49 1234 5678
```

Herr Mustermann soll nun eine Telefonnummer erhalten. LDAP sieht hierfür das Attribut *telephoneNumber* vor. Der fiktive Wert soll *+49 1234 5678* lauten, Listing 8 zeigt die passende LDIF-Datei. Die MODIFY-Operation führt das folgende Kommando durch:

```
ldapmodify -x -H ldap://127.0.0.1 -D cn=root,
 o=tutorial -w P@ssw0rd -f listing-8.ldif
```

Hat das geklappt, sollen als Nächstes eine E-Mail-Adresse (Attribut *mail*) sowie eine Ortsangabe zu Herrn Mustermann im LDAP gespeichert werden. Dabei ist dem Administrator unklar, welches Attribut sich für den Ort (englisch: „Locality“) eignet. Aus Listing 4 entnimmt man, dass das Schema des OpenLDAP-Servers unter der Suchbasis *cn=subschema* per LDAP abfragbar ist. Diese Funktion erweist sich bei der Suche nach geeigneten Attributen als äußerst praktisch. Ein *ldapsearch* mit den Parametern *–b cn=subschema –s base* + listet auch sämtliche Attributdefinitionen. Durchforscht man diese nach dem String „locality“, findet man die folgende Definition:

```
attributeTypes: ( 2.5.4.7
NAME ( 'l' 'localityName' )
DESC 'RFC2256: locality which this
 object resides in'
SUP name )
```

Das passende Attribut für den Ort lautet also einfach *l*. Der Versuch, eine zu Listing 8 analoge LDIF-Datei mit *l: Musterstadt* zu erzeugen und via *ldapmodify* anzuwenden, scheitert allerdings. Das Tool quittiert den Versuch mit den Meldungen „Object class violation“ und „attribute ‚l‘ not allowed“. Das liegt schlichtweg daran, dass die in Listing 6

für Herrn Mustermann verwendete Objektklasse *person* dieses Attribut nicht vorsieht.

Listing 9: Auslesen des Schemas des LDAP-Servers

```
# ldapsearch -H ldap://127.0.0.1 -x -LLL -b cn=subschemas base
+
[...]
objectClasses: ( 2.5.6.6
 NAME 'person'
 DESC 'RFC2256: a person'
 SUP top
 STRUCTURAL
 MUST ( sn $ cn )
 MAY ( userPassword $ telephoneNumber $ seeAlso $ description
) )

objectClasses: ( 2.5.6.7
 NAME 'organizationalPerson'
 DESC 'RFC2256: an organizational person'
 SUP person
 STRUCTURAL
  MAY ( title $ x121Address $ registeredAddress $
destinationIndicator $
    preferredDeliveryMethod $ telexNumber $
teletexTerminalIdentifier $
   telephoneNumber $ internationaliSDNNumber $
facsimileTelephoneNumber $
 street $ postOfficeBox $ postalCode $ postalAddress $
 physicalDeliveryOfficeName $ ou $ st $ l ) )

objectClasses: ( 2.16.840.1.113730.3.2.2
 NAME 'inetOrgPerson'
 DESC 'RFC2798: Internet Organizational Person'
 SUP organizationalPerson
 STRUCTURAL
  MAY ( audio $ businessCategory $ carLicense $
departmentNumber $ displayName $
 employeeNumber $ employeeType $ givenName $ homePhone $
homePostalAddress $
 initials $ jpegPhoto $ labeledURI $ mail $ manager $ mobile $
o $ pager $
```

```
 photo $ roomNumber $ secretary $ uid $ userCertificate $
x500uniqueIdentifier $
 preferredLanguage $ userSMIMECertificate $ userPKCS12 ) )
```

Durchsucht man das Schema nach weiteren Objektklassen, so findet man neben etlichen anderen auch die in Listing 9 dargestellten. Bei der Definition der Objektklasse *organizationalPerson* findet man in den unter dem Strichwort MAY gelisteten Attributen auch *l* – Objekte dieser Klasse können also einen Ort haben. *SUP person* bedeutet, dass *organizationalPerson* von der Klasse *person* abgeleitet ist. Daher erbt es deren Eigenschaften und kann diese um weitere (wie *l*) ergänzen.

Mit dem in der Definition enthaltenen Schlüsselwort *STRUCTURAL* hat es Folgendes auf sich: LDAP unterscheidet verschiedene Typen von Klassen: Die Klasse *top*, von der sich alle anderen ableiten, ist vom Typ „abstract“, die meisten anderen Klassen sind entweder „structural“ oder „auxiliary“. Ein Objekt kann nur solchen strukturellen Objektklassen angehören, die auseinander oder aus der Klasse *top* abgeleitet wurden. *organizationalPerson* ist vom Typ „structural“, ebenso wie die Klasse *person* aus der sie abgeleitet wurde. Die strukturelle Klasse eines Objektes lässt sich per LDAP-Modify-Operation nachträglich nicht mehr ändern. Die bisherigen Benutzereinträge sind daher zu löschen und neu anzulegen. Vorher sollte man sich aber genau überlegen, welche strukturellen Objektklassen zum Einsatz kommen sollen. Spätestens wenn es um das Speichern der E-Mail-Adresse *m.mustermann@tutorial.org* geht, zeigt sich nämlich, dass *organizationalPerson* auch noch nicht der Weisheit letzter Schluss ist. Wie man Listing 9 entnehmen kann, fehlt in dessen Definition nämlich das Attribut *mail*. Dieses findet sich aber in der Liste der MAY-Attribute der Klasse *inetOrgPerson*.

Listing 10: Vollständiges Objekt für Herrn

Mustermann

```
dn: cn=Max Mustermann,ou=people,o=tutorial
changetype: delete

dn: cn=Max Mustermann,ou=people,o=tutorial
changetype: add
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Max Mustermann
sn: Mustermann
givenName: Max
initials: MM
displayName: Max Mustermann
description: Eine Beispielperson
jpegPhoto::
/9j/4AAQSkZJRgABAQEASABIAAD/2wBDABUOEBIQDRUSERIYFhUZHzQiHx0dH0
AuMCY
0TENQT0tDSUhUXnlmVFlyWkhJaY9qcnyAh4iHUWWUn5ODnXmEh4L/wgALCAAcA
BYBAREA/8QAFgABA
QEAAAAAAAAAAAAAAAAABAUD/9oACAEBAAAAAZ9cRq2EmyQaMf/EAB8QAAICAQQ
DAAAAAAAAAAAAAE
CAwQAERMhMiIxQf/aAAgBAQABBQKMAvtRAWEVcT1F5xWuv2t1sMC+mI5WMDn/x
AAcEAACAgIDAAAAA
AAAAAAAAAAAAQIQIUEREiL/2gAIAQEABj8C9GUJwGJ7RHNYkdVqnxuv/8QAHRA
BAAMAAwADAAAAAAAA
AAAAAAQARITFBUWGBkf/aAAgBAQABPyGhLL9nHX7NfA47HWp9kKsnQ1LhNm4Op
Zpxc3DCWHX5ibYGu
ra9w/U//9oACAEBAAAAECev/8QAGhABAQEBAQEBAAAAAAAAAAAAAAAREAIUExUf/
aAAgBAQABPxA4mvU
aAiH1WF6xCnuDWccigv5nftdLVLvun7q7xHgJ5mJ
VM/d//Z
mail: m.mustermann@tutorial.org
telephoneNumber: +49 1234 5678
street: Musterstr. 21
st:: TXVzdGVybMOkbmRsZQo=m
l: Musterstadt
```

Ein entsprechend vervollständigtes Objekt für Herrn Mustermann mit einer Reihe weiterer praktischer Attribute zeigt das

Listing 10. Das Objekt muss man wegen der Änderung der strukturellen Objektklasse löschen und neu anlegen. Es sollte sich nun problemlos mit *ldapmodify* anwenden lassen. Die Namen der Attribute sind selbsterklärend, interessanter ist hier die LDIF-Syntax: Im einfachsten Fall besteht eine Zeile aus einem Attributnamen, gefolgt von einem „*:*“ und dem zugehörigen Wert. „*::*“ bedeutet, dass der Wert Base64-kodiert ist, da LDIF nur 7-Bit-ASCII-Zeichen enthalten darf. Im Beispiel enthält der Wert des Attributes *st* (kurz für *stateOrProvinceName*) aber einen Umlaut; ebenso ist das binäre JPEG-Foto in Base64 einzutragen. Umbrüche sind in LDIF innerhalb eines Wertes ebenfalls erlaubt, dazu rückt man die Folgezeile mit einem Leerzeichen ein, wie das Beispiel zeigt.

Listing 11: LDIF-Datei zum Ändern mehrerer Attribute

```
dn: cn=Max Mustermann,ou=people,o=tutorial
changetype: modify
replace: street
street: Musterstr. 21
-
delete: jpegPhoto
-
add: userPassword
userPassword: SSHAs+o/41d+BW9fQUuQRzf9RtqpcMEDhB78
```

Mit LDIF lassen sich auch Attribute ändern: Es sei beispielsweise die Hausnummer 21 im Attribut *street* falsch und soll nach 12 geändert werden. Das Foto ist zu groß und soll gelöscht werden. Herr Mustermann soll LDAP auch für die Authentifizierung nutzen und benötigt daher ein Passwort. All diese Änderungen lassen sich wie in Listing 11 dargestellt anwenden. Den Hash für das *userPassword* erzeugt man mit dem Kommando *slappasswd*.

Fazit

Dieser erste Teil des Tutorials hat die Grundlagen von LDAP

insbesondere im Hinblick auf das Datenmodell beleuchtet. Die DIT-Struktur für das Tutorial besteht aus einer Organisation (*o=tutorial*) und einer Reihe darunter angeordneter Organisationseinheiten. In der *ou=people* liegen bereits zwei Personenobjekte. Über Suchoperationen können Administratoren und Anwender nach Informationen in diesen Daten zu suchen. Zum Abfragen von Server-Eigenschaften wie dem Schema haben sich LDAP-Suchoperationen ebenfalls als nützlich erwiesen.

Mit LDIF-Dateien und den OpenLDAP-Werkzeugen lassen sich verschiedene Operationen am LDAP-Datenbestand durchführen. Bislang ist die Konfiguration des *slapd* noch sehr rudimentär und bedarf dringend einer Überarbeitung. Teil 2 des Tutorials vertieft daher die *slapd*-Konfiguration im Hinblick auf dynamische das Konfigurations-Backend. Dort werden die in Teil 1 erlernten LDAP-Operationen dann direkt auf Konfigurationsobjekte angewandt. (avr) Mark Pröhl ist Senior Solutions Architect bei der Tübinger Atos-Tochter science + computing AG.

Schnell nachgeschlagen

- OpenLDAP
  1
- LDAP Tool Box Project
  2
- RFC-4512
  3
- RFC-2849
  4

[/expand]