2. Auflage

Stefan **KANIA**
Andreas **OLLENBURG**

OpenLDAP in der Praxis

Das Handbuch für Administratoren

Basierend auf Version 2.6

HANSER

Kania / Ollenburg

OpenLDAP in der Praxis

Ihr Plus – digitale Zusatzinhalte!

Auf unserem Download-Portal finden Sie zu diesem Titel kostenloses Zusatzmaterial.

Geben Sie auf **plus.hanser-fachbuch.de** einfach diesen Code ein:

plus-Kre7f-B34mb

Bleiben Sie auf dem Laufenden!

Unser **Computerbuch-Newsletter** informiert Sie monatlich über neue Bücher und Termine. Profitieren Sie auch von Gewinnspielen und exklusiven Leseproben. Gleich anmelden unter:

www.hanser-fachbuch.de/newsletter

Stefan Kania
Andreas Ollenburg

OpenLDAP in der Praxis

Das Handbuch für Administratoren

2., aktualisierte Auflage

HANSER

Die Autoren:
Stefan Kania, St. Michaelisdonn
Andreas Ollenburg, Dörentrup

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Lektorat: Brigitte Bauer-Schiewek
Copy editing: Jürgen Dubau, Freiburg/Elbe
Layout: le-tex publishing services GmbH
Umschlagdesign: Marc Müller-Bremer, www.rebranding.de, München
Umschlagrealisation: Max Kostopoulos
Titelmotiv: © Max Kostopoulos
Druck und Bindung: Hubert & Co. GmbH & Co. KG BuchPartner, Göttingen
Printed in Germany

Print-ISBN: 978-3-446-47795-7

E-Book-ISBN: 978-3-446-47835-0

E-Pub-ISBN: 978-3-446-47965-4

Inhalt

Vorwort

Herzlich willkommen bei der zweiten Auflage unseres OpenLDAP-Buchs. Es ist endlich soweit, die lange erwartete neue Version von OpenLDAP ist erschienen. Okay, das war schon im Mai 2021, aber bis die Pakete verfügbar waren und die ersten Versionen ihre Kinderkrankheiten durchlaufen haben, wollten wir mit der neuen Auflage warten.

Was Sie hier in der Hand halten, ist eine komplett neu überarbeitete Auflage mit vielen Neuerungen und Erweiterungen. Wer von Ihnen die erste Auflage kennt, wird feststellen, dass gut 100 Seiten dazugekommen sind, und das, obwohl wir alles, was die statische Konfiguration betrifft, aus dem Buch entfernt haben. Wir haben uns ausschließlich auf die dynamische Konfiguration beschränkt. Warum? Einige Funktionen wie zum Beispiel die neu überarbeitete Replikation machen mit der statischen Konfiguration einfach keinen Sinn mehr – im Gegenteil: Sie schränkt dort an einigen Stellen sogar ein.

Es gibt auch neue Kapitel, die zwar von der Thematik her nicht unbedingt neu sind, aber doch überarbeitet wurden. Dazu gehören die Themen OpenLDAP-Proxy, Einrichtung von Referrals, Einrichtung von OpenLDAP über Ansible und last but not least das neue Modul *lloadd*, was aus einem OpenLDAP-Server einen Loadbalancer speziell für OpenLDAP bereitstellen kann.

Das Kapitel der ACLs wurde auch gründlich überarbeitet und um das Thema *sets* erweitert. Auch das Thema Sicherheit spielt wieder eine große Rolle. Wie schon in der ersten Auflage haben wir auch dieses Mal wieder ein ausführliches Kapitel zum Thema Kerberos im Buch, bei Kerberos selbst hat sich nicht so viel getan, aber wir haben die Konfiguration noch etwas genauer erklärt.

Wir verwenden im Buch ausschließlich den neuen Passwordhash ARGON2 und erklären auch, wie Sie eine Zwei-Faktor-Authentifizierung mit OpenLDAP realisieren können.

Mit der neuen Version ist noch die Möglichkeit der automatischen Erstellung von Client-Zertifikaten für Benutzer und Hosts als neue Funktion vorhanden, auch das ist Thema dieser Auflage.

Wie bei anderen Fachbüchern auch lebt eine neue Auflage auch von konstruktivem Feedback der Leser. Auch wir haben davon profitiert.

Wir hoffen, dass wir Ihnen mit dem Buch helfen können, einen guten Einstieg zu bekommen oder eine bestehende alte Installation auf den aktuellen Stand zu bringen. Für uns war es wieder wichtig, dass Sie die einzelnen Beispiele auch praktisch nachvollziehen können.

Alle LDIF-Dateien und Konfigurationen erhalten Sie auf

https://www.kania-online.de/wp-content/uploads/2023/04/listings.zip

oder auf der Downloadseite des Verlages: Geben Sie dazu unter

https://plus.hanser-fachbuch.de

den Code `plus-Kre7f-B34mb` ein.

Vorwort von Stefan Kania

Ich schreibe jetzt schon seit über zehn Jahren Fachbücher. Angefangen hat alles mit dem Linux-Server-Buch. In dem Buch habe ich auch die beiden Kapitel zum Thema OpenLDAP und Kerberos verfasst. Fast genauso lange habe ich immer wieder überlegt, ein Buch nur zum Thema OpenLDAP zu schreiben, in dem ich alle meine Erfahrungen zusammenfassen kann und genügend Platz habe, alle meine Ideen umzusetzen. Das habe ich dann, zusammen mit Andreas, 2020 auch umgesetzt. Uns war klar, dass es erst eine zweite Auflage geben kann, wenn eine neue OpenLDAP-Version erschienen und stabil ist. So war ich dann auch Feuer und Flamme und konnte auch Andreas wieder überreden, eine neue Auflage zu schreiben. Ich hätte gerne noch früher damit angefangen, aber dann habe ich doch etwas auf die Bremse getreten, um die neue Version etwas genauer zu testen.

Danksagungen

Danken möchte ich natürlich wieder Andreas, dass er sich hat breitschlagen lassen, noch einmal mit mir ein Buch zu veröffentlichen. Danke auch an den Hanser Verlag für das Vertrauen und die Unterstützung.

Vieles, was wir in diesem Buch erklären, habe ich bei einem langjährigen Kunden schon sehr früh geplant und umgesetzt und konnte so noch schneller damit beginnen, diese Auflage zu erstellen. Für das Vertrauen möchte ich auch meinen Dank in diese Richtung senden. Wenn ich jetzt „Danke, Andreas“ sage, ist damit nicht mein Co-Autor gemeint, sondern der Verantwortliche für die Umstellung beim Kunden.

Vorwort von Andreas Ollenburg

Ich war lange Jahre als Trainer und Consulter für SUSE-, Novell- und Microsoft-Produkte auf Achse. Dadurch war das Thema „Verzeichnisdienst“ vor allem durch die Arbeit mit dem liebgewonnenen NDS/eDirectory, aber auch mit Active Directory, X.500- und eben auch OpenLDAP-Verzeichnissen immer präsent und entwickelte sich im Lauf der Zeit zu einem meiner Schwerpunkte. Auf meine Zertifizierung zum „Certified Directory Engineer“ von Novell (die Älteren unter uns werden sich erinnern) bin ich auch heute noch ein bisschen stolz.

Nach wie vor ist es hinsichtlich OpenLDAP nicht immer einfach, einheitliche Dokumentationen zu finden. Als wir uns 2019 in Berlin wieder einmal trafen und Stefan mir dieses Buchprojekt zum ersten Mal vorstellte, wurde ich daher schnell hellhörig. Bis er mich dann aber soweit hatte, dass ich mich mit ihm an ein Buch wagen wollte, bedurfte es dann tatsächlich noch etwas Überzeugungsarbeit – nebst dem gemeinsamen Genuss geistvoller Getränke. Aber ich war dann doch froh, mich von Stefan habe überreden zu lassen.

Und jetzt geht unser gemeinsames Erstlingswerk also in die zweite Runde. Als Stefan sich Ende 2022 bei mir meldete und mir eine neue Auflage des Buches schmackhaft machte, konnte ich nicht Nein sagen. Denn wer Stefan kennt, der weiß, wie charmant überzeugend er sein kann. Und was soll man auch sonst an langen und dunklen Winterabenden Besseres tun?

Danksagungen

Mein erster Dank geht natürlich an Stefan. Die Zusammenarbeit hat wieder mal hundertprozentig geklappt und wie immer Spaß gemacht. Darüber hinaus hatte er immer wieder wertvolle Tipps, und wir konnten gemeinsam viele gute Ideen entwickeln. Die Anzahl unserer grauen Haare hat sich wahrscheinlich wieder etwas vergrößert, aber das war es wieder wert.

Bezogen auf den Hanser Verlag kann ich mich Stefans Worten nur anschließen.

Der größte Dank geht aber auch dieses Mal wieder an die drei wertvollsten Menschen in meinem Leben: der besten Ehefrau von allen – auch wenn sie dabei wieder mit den Augen rollt – und unsere beiden großartigen, ebenfalls natürlich besten Kinder. Das Schreiben hat den Dreien wieder einiges an Geduld und Rücksicht abgefordert. Und wenn es mal wieder so gar nicht laufen wollte, wenn sich z. B. ein Fehler im Satzprogramm versteckt hatte oder ich das fehlende Leerzeichen in einer Konfiguration nicht finden konnte, haben sie wieder viel Geduld aufgebracht, mir den Rücken freigehalten oder mich motiviert. Daher also auch diesmal wieder: „Danke, ihr Drei, was wäre ich ohne Euch?“

1 Einleitung

An dieser Stelle möchten wir Ihnen die verschiedenen Formatierungsmöglichkeiten und Administrationsarten erklären. Hier finden Sie auch die Beschreibung zu den im Buch verwendeten Icons, Distributionen und OpenLDAP-Versionen.

1.1 Formales

Damit Sie den größtmöglichen Nutzen aus diesem Buch ziehen können, sollen im Folgenden einige Konventionen erläutert werden.

Kommandozeile vs. grafische Administration

An den meisten Stellen im Buch verwenden wir die Kommandozeile, um die Dienste zu konfigurieren oder zu testen, aber auch die Maus kommt zum Einsatz, um Ihnen auch eine unterschiedliche Auswahl an grafischen Werkzeugen zu zeigen.

In diesem Buch geht es um OpenLDAP und die Verwaltung und Einrichtung des Dienstes, aber auch dem Thema Kerberos ist ein großes Kapitel gewidmet, denn LDAP erhöht zusammen mit Kerberos die Sicherheit Ihrer Umgebung.

1.2 Schriftarten

Viele der Beispiele zu den Kommandos werden in Listings dargestellt. In den Listings werden Sie von der Befehlszeile bis zum Ergebnis alles nachvollziehen können, wie Sie hier im Beispiel sehen:

Listing 1.1 Ein Test-Listing

```
stefan@samba4~\$ ps
 PID TTY          TIME CMD
 4008 pts/2    00:00:00 bash
 4025 pts/2    00:00:00 ps
```

Um Kommandos, Dateinamen und Konfigurationen im Text besser hervorzuheben, werden die folgenden Schriftarten verwendet:

- Um bestimmte Begriffe hervorzuheben, werden sie *schief* geschrieben.
- Für die Darstellung von Tastenkombinationen und Klicks auf bestimmte Symbole oder Karteireiter in der grafischen Oberfläche werden KAPITÄLCHEN verwendet.
- Wenn im Text der Hinweis auf eine Datei gegeben wird, werden wir die Schriftart Sans Serif verwenden.
- Im Fließtext werden Konsolenbefehle mit der Schrift `Schreibmaschine` geschrieben.
- Parameter und Werte aus Listings werden durch die Verwendung von *Kursivschrift* gekennzeichnet.

1.2.1 Eingabe langer Befehle

Es gibt noch eine weitere wichtige, eher technische Konvention: Einige der vorgestellten Kommandozeilenbefehle oder Ausgaben von Ergebnissen erstrecken sich über mehrere Buchzeilen. Im Buch kennzeichnet am Ende der entsprechenden Zeilen ein „\“, dass der Befehl oder die Ausgabe in der nächsten Zeile weitergeht. Achten Sie besonders darauf, wenn Sie Kommandos aus dem Buch abtippen.

1.2.2 Screenshots

Die Verwendung von Screenshots zur Erhellung von Sachverhalten werden wir im Laufe des Buches gezielt einsetzen. Oft geht es darum, Hierarchien besser darstellen zu können oder die Verwendung von grafischen Werkzeugen zu verdeutlichen.

1.2.3 Internetverweise

An einigen Stellen werden wir auf bestimmte URLs verweisen – sei es, um Ihnen Quellen für bestimmte Downloads zu geben, oder um Ihnen den Weg zu tiefer gehenden und weiterführenden Erklärungen zu geben, die den Rahmen dieses Buches sprengen würden. Verweise auf Internetadressen werden immer vollständig angegeben, zum Beispiel so: *https://www.openldap.org*.

1.2.4 Icons

Sie werden in den einzelnen Kapiteln am Rand oft Icons finden, die Sie auf bestimmte Zusammenhänge oder Besonderheiten hinweisen sollen. Die Icons haben die folgenden Bedeutungen:

Wichtig: Wann immer Sie das nebenstehende Symbol sehen, ist Vorsicht angeraten: Hier weisen wir auf besonders kritische Einstellungen hin oder auf Fehler, die dazu führen können, dass das System nicht mehr stabil läuft. Damit sich die Warnungen mehr vom übrigen Text abheben, werden diese Textbereiche dann noch mit einem grauen Kasten hinterlegt.

Hinweis: Alle Textstellen mit diesem Icon sollten Sie unbedingt lesen! Hier handelt es sich oft um wichtige Hinweise, die Sie nicht außer Acht lassen sollten.

Tipp: Bei diesem Symbol finden Sie nützliche Tipps und Tricks zu bestimmten Aufgaben.

1.3 Linux-Distributionen

Für die Verwendung von OpenLDAP ist es nicht sehr relevant, welche Distribution Sie einsetzen. Da wir aber nicht die Pakete aus den Distributionen einsetzen (denn die basieren meist noch auf OpenLDAP 2.4), spielen auch unterschiedliche Pfade auf unterschiedlichen Distributionen keine Rolle mehr. Wir werden ausschließlich die Pakete der Firma Symas einsetzen, da nur diese Pakete immer aktuell sind. Warum die Pakete der Firma Symas? Die Entwickler vom OpenLDAP haben diese Firma gegründet und stellen somit auch immer die aktuellsten Pakete bereit. Egal welche Distribution Sie einsetzen, die Symas-Pakete installieren sich immer in das Verzeichnis /opt. Ein Unterschied besteht nur in der eigentlichen Installation der Pakete.

Beim Thema Kerberos werden wir Debian 11 einsetzen. Da sich zu den Redhat-basierten Distributionen der Pfad zu den Konfigurationsdateien ändert, werden wir an der Stelle auf die unterschiedlichen Pfade hinweisen. Aber auch die Funktionsweise von Kerberos ist bei allen Distributionen identisch, sodass die Wahl der Distribution hier auch keine Rolle spielt.

Im Buch werden wir auch auf die Einbindung von Kerberos in OpenLDAP eingehen. Um alle Beispiele für alle Distributionen einheitlich zu halten, werden wir ausschließlich den MIT-Kerberos einsetzen, da Sie diesen, im Gegensatz zum Heimdal-Kerberos, auf allen Distributionen finden.

Ein Hinweis zu Firewalls, SELinux und Apparmor: Wir werden vor der Installation diese Systeme immer deaktivieren, da es in diesem Buch nicht um das Thema Systemsicherheit geht.

1.4 Downloads zum Buch

Alle hier im Buch verwendeten Listings, LDIF-Dateien und Skripte stellen wir Ihnen unter der URL *https://www.kania-online.de/wp-content/uploads/2023/04/listings.zip* zur Verfügung.

1.5 OpenLDAP-Version

Mit der Einführung der neuen OpenLDAP-Version gibt es jetzt zwei verschiedene Zweige, die gepflegt werden. Zum einen ist da die Version 2.5, dabei handelt es sich um eine LTS-Version, die über einen längeren Zeitraum mit Updates versorgt wird, aber nicht alle neu entwickelten Funktionen erhalten wird. Zum anderen gibt es die Version 2.6, die auf der 2.5 aufsetzt, aber immer wieder neue Funktionen bereitstellt. Hier ist es Ihre Entscheidung, welchen Weg Sie gehen wollen. Wir werden hier die Version 2.6 einsetzen. Wir werden versuchen, Sie immer auf die Funktionen hinzuweisen, die nur in der Version 2.6 vorhanden sind. Die Unterschiede finden Sie aber auch immer in den Release Notes zu den Versionen.

1.6 Konfigurationsarten

OpenLDAP lässt sich auf zwei verschiedene Arten konfigurieren. Da wäre einmal die Möglichkeit über die statische Konfiguration mithilfe der Datei slapd.conf und die Möglichkeit der dynamischen Konfiguration über das eigene Backend *cn=config*. In der ersten Auflage des Buches haben wir immer beides gezeigt, aber auch schon erwähnt, dass die statische Konfiguration als *deprecate* gekennzeichnet ist. Aus diesem Grund werden wir hier nur noch auf die dynamische Konfiguration eingehen. Lediglich am Anfang werden wir die Grundkonfiguration über beide Wege erklären, um Ihnen zu zeigen, dass Sie alle Einträge aus der slapd.conf auch im Backend *cn=config* wiederfinden.

Suchen Sie die Schreibweise für bestimmte Parameter für die dynamische Konfiguration, dann finden Sie eine gute Tabelle, die beide Parameter (statisch) und (dynamisch) gegenüberstellt, unter *https://tylersguides.com/guides/openldap-online-configuration-reference/*.

Jetzt bleibt uns nur noch, Ihnen viel Spaß mit dem Buch zu wünschen und zu hoffen, dass Ihnen das Buch bei Ihrer täglichen Arbeit eine Hilfe sein wird.

2 LDAP-Grundlagen

Bevor wir mit der Einrichtung des ersten OpenLDAP-Servers beginnen, wollen wir Ihnen an dieser Stelle ein paar Grundlagen zum *Lightweight Directory Access Protocol* (LDAP) vermitteln. Mithilfe dieser Grundlagen ist es für Sie später einfacher, bestimmte Zusammenhänge zu verstehen, da Sie dann wissen, wie das Protokoll aufgebaut ist und was es bedeutet, wenn über *Objects*, *Attribute* und *Schemata* gesprochen wird.

LDAP ist ursprünglich kein Dienst, sondern wie es die Abkürzung schon sagt, ein Protokoll, das als Proxy zwischen den TCP/IP-Clients und den DAP-Datenbanken auf Großrechnern eingesetzt wurde. Da diese Großrechner nicht das OSI-Referenzmodell für ihre Protokolle verwendet haben, musste für den Zugriff von Clients aus dem TCP/IP-Netzwerk ein Übergang geschaffen werden, um die Zugriffe weiterleiten zu können. Dafür wurde LDAP entwickelt. Im Laufe der Zeit wurde dann ein eigenes Datenbank-Backend für diesen Proxy-Dienst entwickelt, und daraus ist dann der heutige Serverdienst LDAP entstanden.

LDAP ist ein hierarchisch gegliederter Verzeichnisdienst, und mit seiner Hilfe können Sie Ihre Unternehmensstruktur direkt abbilden. Sie können aber genauso eine Struktur erstellen, die verschiedene Ressourcen zusammenfasst. Sie müssen sich auf jeden Fall immer Gedanken über Ihre Struktur machen, denn eine allgemeingültige Struktur für einen LDAP-Baum gibt es nicht. Planen Sie gut, bevor Sie eine neue Struktur aufbauen. Eine genaue Planung kann Ihnen hinterher sehr viele nachträgliche Anpassungen und Änderungen ersparen. Gerade wenn es später darum geht, Berechtigungen für Zugriffe auf Objekte zu geben, ist die Planung der Struktur sehr wichtig. Ohne eine gute Planung kann es später schwer werden, die Berechtigungen mit wenigen einfachen Regeln zu setzen. Gruppieren Sie Ihre Ressourcen so, dass Sie später die einzelnen Bereiche des Baums gezielt mit Rechten für Gruppen und Benutzer vergeben können. Vergleichen können Sie die Struktur in etwa mit der Struktur Ihrer Dateisysteme auf Fileservern.

2.1 Grundlagen zum Protokoll

Die Entwicklung des *Lightweight Directory Access Protocol* stammt aus dem Jahr 1993. Es wurde benötigt, um den Zugriff auf DAP-Datenbanken über TCP/IP zu erleichtern. Der ursprüngliche X.500-Standard, der für DAP-Datenbanken verwendet wurde, umfasst alle sieben Ebenen des OSI-Referenzmodells. Das machte eine Implementation auf verschiedene

Systeme aufwendig oder gar unmöglich. Aus diesem Grund wurde im Jahre 1993 das Protokoll LDAP entwickelt. Am Anfang wurde es nur verwendet, um auf DAP-Server zugreifen zu können. LDAP diente dabei mehr oder weniger als Proxy, um zwischen X.500 und den verschiedenen Systemen zu vermitteln. Der große Vorteil von LDAP gegenüber einer reinen DAP-Umgebung ist der, dass für LDAP nur ein funktionsfähiger TCP/IP-Protokollstack benötigt wird. Später wurde zu LDAP ein eigenes Datenbank-Backend hinzugefügt, um unabhängig von den DAP-Servern zu werden. Heute wird LDAP in verschiedenen Produkten als Verzeichnisdienst eingesetzt. Dazu gehören unter anderem Microsoft Active Directory, OpenLDAP oder der 389 Directory Server von Red Hat.

Wie schon erwähnt, handelt es sich bei LDAP um einen Verzeichnisdienst. Ein Verzeichnisdienst zeichnet sich durch die folgenden Eigenschaften aus:

- Die Administration kann auf verschiedene Bereiche aufgeteilt werden.
- Die gesamte Unternehmensstruktur kann 1:1 im Verzeichnis abgebildet werden.
- Durch eine Partitionierung kann die Struktur auf mehrere Server verteilt werden.
- Für die Suche nach bestimmten Objekten im Verzeichnis können Sie mehr oder weniger komplexe Filter einsetzen, die die Suche auf bestimmte Teilbereiche des Verzeichnisses einschränken und dadurch die Suche beschleunigen.
- Da LDAP auf X.500 basiert, wird hier ein objektorientiertes Datenmodell verwendet. Dadurch ist eine Vererbung von Eigenschaften auf andere Objekte möglich.

Für alle Verzeichnisdienste existiert der X.500-Standard der *ITU-T* der internationalen Fernmeldeunion *https://www.itu.int*. Diese Standards beschreiben, wie Verzeichnisdaten zur Verfügung gestellt und abgerufen werden und wie die Verschlüsselung, Authentifizierung, Replikation und Verwaltung der Verzeichnisdaten gehandhabt werden. Die X.500-Standards liefern die Funktionsmodelle und Begriffsbestimmungen für die Verzeichnisdienste, die nicht voll auf dem X.500-Standard basieren, was bei LDAP der Fall ist.

Im Laufe seiner Entwicklung wurde LDAP immer weiterentwickelt, und die aktuelle Version von LDAP ist die Version 3. Verwechseln Sie die LDAP-Version nicht mit der Version der Produkte wie zum Beispiel OpenLDAP Version 2.6. Die Versionsnummer 2.6 bezieht sich hier auf die Version der Software und nicht auf die Version des Protokolls. Die aktuelle Version 3 von LDAP hat gegenüber der Version 2 die folgenden Vorteile:

- Authentifizierung durch Verwendung von SASL für die Verschlüsselung der Passwörter. Hier wird heute hauptsächlich Kerberos eingesetzt. Im Buch wird es dazu ein eigenes Kapitel geben.
- Verschlüsselung des gesamten Datenverkehrs im Netz durch TLS.
- Möglichkeit der Verwendung von *UTF-8* für Attribute.
- Verweise auf andere LDAP-Server, die *Referrals*. Dadurch können Ressourcen in mehreren Bäumen gemeinsam genutzt, aber getrennt administriert werden.

Sollten Sie heute noch alte LDAPv2-Server im Netz finden, können Sie diese nicht mit LDAPv3-Servern replizieren, da die beiden Protokolle nicht kompatibel zueinander sind. Es ist aber immer noch möglich, ältere Software, die nur mit LDAPv2-Client-Software ausgestattet ist, mit einem LDAPv3-Server zu verbinden. Wie lange das noch möglich ist, bleibt aber fraglich. Sollten Sie noch Software nutzen, die einen alten LDAPv2-Client nutzt, sollten Sie diese möglichst bald aktualisieren.

2.1.1 Der Einsatz von LDAP im Netzwerk

Sie können LDAP auf verschiedene Arten in Ihrem Netzwerk für die Verwaltung einsetzen. Aber erst zusammen mit Kerberos ist es auch möglich, ein *Single Sign-on* im Netzwerk zu realisieren. Ohne Kerberos können Sie lediglich eine zentrale Benutzerverwaltung aufbauen, die Ihre Anwender dann für die Anmeldung an verschiedenen Diensten nutzen können. Aber denken Sie daran: Sämtliche Datenübertragung zwischen einem LDAP-Server und einem LDAP-Client läuft immer erst einmal unverschlüsselt. Erst durch den Einsatz von *TLS* oder *LDAPS* werden die Daten sicher übertragen.

An ein LDAP-Verzeichnis können Sie die unterschiedlichsten Dienste anbinden und die Daten des LDAP für unterschiedliche Aufgaben nutzen. In der folgenden Aufzählung finden Sie eine kleine Übersicht der Möglichkeiten:

- Verwaltung von Benutzern, Gruppen für Posix-Konten
- Verwaltung von Weiterleitungen und Aliasen für den Postfix-Mailserver
- Einrichtung der Benutzerauthentifizierung für einen IMAP-Server
- Authentifizierung von Benutzern für die Anmeldung bei einem Proxy
- Authentifizierung von Benutzern für die Anmeldung am Webserver

2.1.2 Das LDAP-Datenmodell

Das Datenmodell von LDAP ist objektorientiert, d. h. einzelne Objekte setzen sich aus Objektklassen und Attributen zusammen. Bei LDAP gelten dabei fast die gleichen Regeln wie bei der objektorientierten Programmierung. Auch im LDAP gibt es Vererbung und Polymorphie. Im Gegensatz zur objektorientierten Programmierung wird hier aber sehr viel Gebrauch von der Polymorphie gemacht, sprich ein Objekt besteht aus mehreren Objektklassen. In Bild 2.1 sehen Sie, wie sich ein Objekt aus verschiedenen Objektklassen und Attributen zusammensetzt.

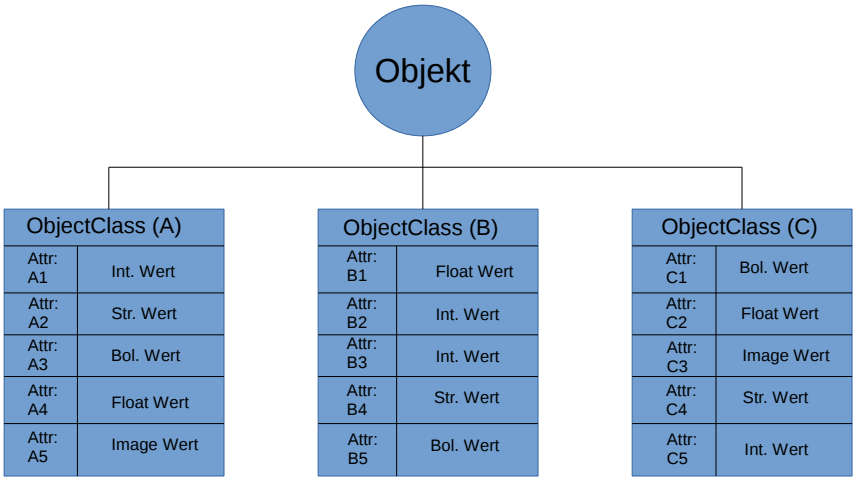

Bild 2.1 Aufbau eines Objekts

Die Aufgabe von LDAP ist es, die Objekte abzubilden und miteinander in Beziehung zu bringen. Ein Objekt wird im Verzeichnisbaum als *Verzeichniseintrag* bezeichnet. Jedes Objekt wird über seinen eindeutigen Namen, den *Distinguished Name* (dn), im *Directory Information Tree* (DIT) angelegt, der ähnlich wie der Name einer Datei im Dateisystem behandelt wird. Durch die verschiedenen Objekte entsteht so nach und nach eine Baumstruktur.

Im LDAP-Baum wird zwischen zwei verschiedenen Objektarten unterschieden. Zunächst gibt es die *Organizational Unit* (OU), bei der es sich um ein Containerobjekt handelt. In diesen Organizational Units können weitere Objekte erzeugt werden. Die OUs werden zum Aufbau der Struktur verwendet. Um noch einmal den Vergleich zum Dateisystem zu ziehen: Eine OU können Sie mit den Verzeichnissen im Dateisystem vergleichen, nur dass eine OU immer auch Attribute besitzt.

Die andere Objektart sind die *Blattobjekte*, die zum Beispiel mit der Kennzeichnung *commonName* (cn) oder *users ID* (uid) im Verzeichnisbaum Verwendung finden. Diese Objekte dienen zur Verwaltung der Ressourcen. Die Bezeichnungen werden aus den verschiedenen Objektklassen und Schemata abgeleitet, auf die wir im Verlauf dieses Kapitels noch weiter eingehen werden. Wie Sie gelesen haben, unterstützt die LDAP-Version 3 auch UTF-8-Zeichen. Sie können also für die Benennung von Objekten und Werten auch Umlaute und Leerzeichen nutzen, wenn Sie Attribute anlegen. Nur werden dann die Attribute BASE64-kodiert im LDAP abgelegt und bei einer Suche mit `ldapsearch` auch BASE64-kodiert angezeigt. Verwenden Sie anstelle des Attributes *cn* das Attribut *uid* für die Benennung Ihrer Benutzer, können Sie den Namen nur so abspeichern, wie es auch unter Unix möglich ist, nämlich ohne Leerzeichen und ohne Sonderzeichen. Dann werden Ihnen die Namen der Benutzer auch immer im ASCII-Format angezeigt. Was aber, wenn ein Objekt kein uid-Attribut besitzt? Dann müssen Sie das cn-Attribut verwenden, sollten aber, besonders wenn Sie viel auf der Kommandozeile suchen, auf Leer- und Sonderzeichen verzichten.

Die meisten grafischen Werkzeuge können aber BASE64 direkt umsetzen und lesbar darstellen.

Hinweis: OpenLDAP weist eine Besonderheit auf gegenüber zum Beispiel dem Active Directory. Im OpenLDAP-Baum können Sie unterhalb des Blattobjekts weitere Objekte erzeugen. So ist es möglich, unterhalb eines Benutzers eine weitere OU zu erzeugen, in der dann zum Beispiel das persönliche Adressbuch des Benutzers gespeichert werden kann.

2.1.3 Attribute

Attribute sind die Eigenschaften von Objekten. Ein Attribut besteht aus einem Namen, mit dem das Attribut innerhalb eines Objekts eindeutig referenziert werden kann, und einem eindeutigen Identifier. Attribute können eine unterschiedliche Anzahl von Werten besitzen. Es gibt Attribute, die nur genau einen Wert haben dürfen, und solche, die mehrere Werte haben können. Um die Wiederverwendbarkeit von Attributen in verschiedenen Objektklassen zu ermöglichen, werden Attribute getrennt von Objekten verwaltet, und zwar in Form von Attributtypen. Der *attributetype* enthält die Komponenten wie in Listing 2.1:

Listing 2.1 Beispiel für ein Attribut

```
attributetype ( 1.3.6.1.1.1.1.4 NAME 'loginShell'
        DESC 'The path to the login shell'
        EQUALITY caseExactIA5Match
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

In der ersten Zeile werden der *Object Identifier* (OID) und der Name des Attributs angegeben. Beide Werte müssen im gesamten DIT eindeutig sein.

Im Anschluss folgt eine Beschreibung, die frei formuliert werden kann. Danach gibt es die Möglichkeit, die Gleichheit *EQUALITY* für die Suche nach Attributen festzulegen. Ein Objekt mit dem Attribut *loginShell* würde nur angezeigt, wenn der Suchbegriff exakt, inklusive der Groß- und Kleinschreibung, übereinstimmt. Im Anschluss folgt noch eine Syntaxbeschreibung in Form eines OID. Alle möglichen OIDs können Sie im RFC 2252 nachlesen. Durch den OID wird festgelegt, um was für eine Art von Objekt es sich hier handelt, zum Beispiel um eine Zahl, eine Zeichenkette oder eine andere Syntax. Neben dem Namen, dem OID und der Beschreibung gibt es bei Attributen noch weitere Eigenschaften, die Sie berücksichtigen müssen, wenn Sie die Verwendung planen. Im Beispiel sehen Sie die Zeile, die mit *EQUALITY* beginnt. Über diese Eigenschaft wird festgelegt, ob bei den zugeordneten Werten die Groß- und Kleinschreibung bei der Suche berücksichtigt wird. Diese Eigenschaft wird als *Matching Rules* bezeichnet. Alle Matching Rules finden Sie im RFC 4517.

Die nächste Zeile definiert die *SYNTAX* eines Attributs, dabei wird wieder ein OID verwendet. Die Beschreibung aller OIDs für die SYNTAX finden Sie ebenfalls im RFC 4517. Wenn Sie bei der SYNTAX nicht den zusätzlichen Parameter *SINGLE-VALUE* angeben, können Sie ein Attribut immer mit mehreren Werten belegen. Nicht bei jedem Attribut macht es Sinn, mehrere Werte zu speichern. Hinter der SYNTAX können Sie noch die maximale Länge des Attributs in geschweiften Klammern angeben. Bei der Länge legen Sie fest, wie viele Zeichen des Werts bei einer Suche geprüft und verglichen werden. Die maximale Länge eines Attributs beträgt 65535 Zeichen. Aber wozu dient die SYNTAX? Über die SYNTAX wird festgelegt, mit welcher Art von Werten die Attribute gefüllt werden können. Hier gibt es verschiedene Möglichkeiten: Ein Attribut kann so bestimmt werden, dass Sie nur Integer-Werte eintragen können oder aber Strings oder Bilder. Auch diese Werte sind wieder genau im RFC 4517 definiert.

2.1.4 Objektklassen

Eine Objektklasse im LDAP ist eine Sammlung von vorher definierten Attributen, die dann für die Zusammenstellung der Objekte verwendet wird. Ein Objekt kann sowohl ein Container, in dem weitere Objekte verwaltet werden, als auch ein Benutzer, eine Gruppe oder eine andere Ressource sein. Eines ist bei allen Objekten aber immer gleich: Alle Objekte bestehen aus einer oder mehreren Objektklassen, diese haben Eigenschaften, die *Attribute*. Die Attribute unterscheiden sich je nachdem, um welche Art von Objekt es sich handelt.

Es werden hauptsächlich zwei verschiedene Objektklassen verwendet:

- STRUCTURAL ObjectClass
  Bei der *STRUCTURAL ObjectClass* handelt es sich um die Objektklasse, die ein Objekt

maßgeblich bestimmt. Man kann sie auch als übergeordnete Objektklasse bezeichnen. Einem Objekt muss immer genau eine STRUCTURAL ObjectClass zugeordnet sein. Ein Austauschen dieser Klasse ist nur sehr schwer möglich. Um einem Objekt eine neue STRUCTURAL ObjectClass geben zu können, werden Sie es im Normalfall neu anlegen.

- AUXILIARY ObjectClass
  Die *AUXILIARY ObjectClass* können Sie zusätzlich zur STRUCTURAL ObjectClass zu einem Objekt hinzufügen. Sie können auch Objekte später um weitere AUXILIARY ObjectClasses erweitern oder die Klassen auch entfernen. Denken Sie nur daran, dass eine Klasse auch immer Attribute besitzt, die Sie bei einer späteren Änderung auch hinzufügen oder entfernen müssen, wenn Sie eine AUXILIARY ObjectClass von einem Objekt entfernen.

Um einen ersten Eindruck einer Objektklasse zu bekommen, sehen Sie in Listing 2.2 ein Beispiel für eine *objectclass*. Hier wird auch deutlich, dass bestimmte Attribute vergeben werden **müssen**, während andere vergeben werden **können**.

Listing 2.2 Beispiel für eine Objektklasse

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with POSIX attributes'
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
```

2.1.5 Objekte

Objekte bilden die Ressourcen in Ihrem Verzeichnisbaum ab. Alle Objekte in Ihrem Verzeichnisbaum benötigen einen eindeutigen Namen. Das Attribut, das zur eindeutigen Adressierung des Objekts verwendet wird, ist in den meisten Fällen der *commonName* eines Objekts. Über dieses Attribut wird das Objekt im Verzeichnisbaum verwaltet.

Ein Objekt kann für die Verwaltung der unterschiedlichsten Aufgaben verwendet werden. Für jede Aufgabe benötigt ein Objekt unterschiedliche Attribute, deshalb kann ein Objekt auch aus mehreren Objektklassen zusammengestellt werden. Wenn zum Beispiel für einen Benutzer ein Objekt erstellt werden soll, das für die Anmeldung an einem UNIX-System benötigt wird, muss dem Objekt die Objektklasse *posixAccount* zugeordnet werden, da diese die Attribute für eine erfolgreiche Anmeldung enthält.

Alle Attribute, die hier hinter *MUST* stehen, müssen Sie zwingend vergeben, wenn Sie ein Objekt mit dieser Objektklasse *posixAccount* erstellen wollen. Die Attribute, die in der Zeile *MAY* stehen, sind optional und können von Ihnen vergeben werden.

2.1.6 Schema

Die Objektklassen werden jetzt nicht einzeln dem LDAP-Server zugeordnet, sondern in Gruppen zu einem Schema zusammengefasst. Für Schemata gilt das Gleiche wie für Objektklassen: Die Standardschemata sollten nicht erweitert werden, da es sonst bei einer eventuellen Zusammenführung zweier Bäume zu Konflikten kommt. Wenn Sie eigene Attribute benötigen, sollten Sie immer eine eigene Objektklasse in einem eigenen Schema

erzeugen. Ein eigenes Schema können Sie auch gut in einen beliebigen LDAP-Baum integrieren. Ein wichtiger Punkt bei der Erstellung eines eigenen Schemas ist der OID für die Attribute und Objektklassen. Natürlich können Sie den OID selbst wählen. Was tun Sie, wenn der selbst gewählte OID schon vergeben ist und später eine Zusammenführung gerade mit diesem Baum stattfinden soll? Deshalb sollten Sie Ihren OID immer registrieren. Den eigenen OID können Sie kostenlos unter der URL *http://pen.iana.org/pen/PenApplication.page* registrieren. Mit diesem Formular erhalten Sie einen OID vom Typ 1.3.6.1.4.1.*. Darunter können Sie nun die eigenen Attribute erstellen und nummerieren.

Neben einem eigenen OID sollten Sie für die Namen ein Präfix festlegen, mit dem alle Attributnamen und die Namen Ihrer eigenen Objektklassen im eigenen Schema beginnen. Wollen Sie zum Beispiel ein Schema für die Verwaltung Ihrer Kunden im Unternehmen erstellen, kann jedes Attribut mit dem Firmennamen als Präfix beginnen. Hier im Beispiel verwenden wir das Präfix *stka-*. Ein Beispiel für einen Attributnamen wäre dann *NAME stka-Name*. Listing 2.3 zeigt ein Beispiel für ein eigenes Schema:

Listing 2.3 Beispiel für ein eigenes Schema

```
# Schema for OpenLDAP-Book about Customers
#
# Last change: January  26, 2023
#
# Created by: Stefan Kania <stefan@kania-online.de>
#
# General guideline:
# 1. The language in this file is english
# 2. Every OID in this file must look like this: ns.a.b.c., where
#    ns - the official namespace of the Customer schema:
#         1.3.6.1.4.1.12345
#    a  - Reserved, must always be 1 for the  Customer schema.
#    b  - Entry type (1:attribute, 2:object)
#    c  - Serial number (increased with every new entry)
# 3. Every entry in this file MUST have a "DESC" field, containing a
#    suitable description!
# 4. Attributes are listed in front of objects. All entries must be
#    ordered by their serial number.
# 5. All attributenames must start with 'stka-'
#
# This schema is not depending on other schemas


# Attribute type definitions

attributetype (1.3.6.1.4.1.12345.1.1.1
        NAME 'stka-CustomerName'
        DESC 'Name of the customer'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{100}
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.2
        NAME 'stka-CustomerPhoto'
        DESC 'JPEG photo of the Customer'
```

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.28
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.3
        NAME 'stka-CustomerStreetName'
        DESC 'streetname of the customer'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.4
        NAME 'stka-CustomerZipCode'
        DESC 'Zipcode of the customer'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.5
        NAME 'stka-CustomerCity'
        DESC 'Cityname of the customer'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.6
        NAME 'stka-CustomerCountry'
        DESC 'Countryname from the customer'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41
        SINGLE-VALUE)

attributetype (1.3.6.1.4.1.12345.1.1.7
        NAME 'stka-CustomerPhonenumber'
        DESC 'Official phonenumber from the customer'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50)

attributetype (1.3.6.1.4.1.12345.1.1.8
        NAME 'stka-CustomerHomepage'
        DESC 'Official homepage of the customer'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{255})

attributetype (1.3.6.1.4.1.12345.1.1.9
        NAME  'stka-CustomerMail'
        DESC 'Mailaddress from the customer'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} )
```

```
attributetype ( 1.3.6.1.4.1.12345.1.1.10
        NAME 'stka-GoodCustomer'
        DESC 'If set to true then the customer is good'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE )

# Objectclass definitions

objectclass ( 1.3.6.1.4.1.12345.1.2.1
        NAME 'stka-Customer'
        DESC 'Objectclass to manage all customers'
        SUP top STRUCTURAL
        MUST (stka-CustomerName $ stka-CustomerCity)
        MAY (stka-CustomerPhoto $ stka-CustomerStreetName
             $ stka-CustomerZipCode
             $ stka-CustomerCity $ stka-CustomerCountry
             $ stka-CustomerPhonenumber
             $ stka-CustomerHomepage $ stka-CustomerMail
             $ stka-GoodCustomer))
```

Wichtig: Achten Sie darauf, dass die Zeilen nach *attributetype* und *objectclass* eingerückt sind, sonst kommt es beim Start des LDAP-Servers zu Fehlern.

Jedes der hier verwendeten Attribute kann sowohl über den OID als auch über den Namen eindeutig referenziert werden. Durch die Verwendung eines eigenen Suffixes kann es auch nicht zu Verwechslungen mit Standardattributen kommen. Da hier der Typ *SUP top STRUCTURAL* für die Objektklasse genutzt wird und die Objektklasse von keiner anderen Objektklasse abhängig ist, können Sie ein Objekt erstellen, das nur aus dieser Objektklasse besteht. Wollen Sie eine Objektklasse erstellen, die Sie anschließend zusätzlich zu bestehenden Objekten hinzufügen wollen, dann müssen Sie an dieser Stelle *SUP top AUXILIARY* schreiben. Dadurch kann die Objektklasse auch im Nachhinein anderen Objekten zugeordnet werden. Sie sind dann aber nicht in der Lage, ein Objekt zu erstellen, das nur aus dieser einen Objektklasse besteht.

Sie haben in Ihrem eigenen Schema immer die Möglichkeit, aus den Attributen mehrere Objektklassen mit unterschiedlichen Namen und unterschiedlichen ODIs, aber denselben Attributen zu bauen, und hätten dann beide Möglichkeiten.

Wenn Sie jetzt das Schema zu Ihrer LDAP-Konfiguration hinzufügen, können Sie die Objektklassen wie gewohnt nutzen.

Die komplette Beschreibung, wie Sie ein eigenes Schema aufbauen, finden Sie inklusive aller Syntaxbeschreibungen im RFC 2252.

Immer, wenn Sie ein eigenes Schema erstellen, müssen Sie es auch in die Konfiguration Ihres LDAP-Servers einbinden. Wichtig ist bei der Konfiguration die Reihenfolge der eingebundenen Schemata. Da eine Objektklasse auch Attribute einer anderen Objektklasse nutzen kann und es dabei nicht wichtig ist, in welchem Schema sich die Objektklasse befindet, müssen Sie immer auf die Reihenfolge der Einträge achten. In Kapitel 3, «Installation des ersten LDAP-Servers», werde ich noch genauer auf die Reihenfolge eingehen.

Im OpenLDAP gibt es einige Schemata, die fast immer eingebunden werden. In Tabelle 2.1 finden Sie eine Übersicht über diese Schemata.

Tabelle 2.1 Schemata des OpenLDAP

Name des Schemas	Funktion
core.schema	Standardschema, muss immer eingebunden sein
cosine.schema	Standardattribute der LDAP-Version 3
nis.schema	Hier befinden sich alle Attribute für POSIX-Konten
inetorgperson.schema	Enthält Attribute, die für die Benutzerverwaltung relevant sind

2.1.7 Umwandeln eines Schemas in ein LDIF

Wie schon in der Einleitung zum Buch beschrieben, werden wir in dieser Auflage nur noch auf die dynamische Konfiguration des OpenLDAP eingehen. Daher wollen wir Ihnen an dieser Stelle erklären, wie Sie eine eigene Schemadatei in eine LDIF-Datei umwandeln können, sodass Sie diese dann später auch in Ihren dynamisch verwalteten LDAP-Baum einbinden können. An dieser Stelle werden wir auf Kommandos zugreifen, die erst später erklärt werden. Aber wenn Sie die Umwandlung an dieser Stelle gleich testen möchten, können Sie das mithilfe der Listings realisieren.

Der Ausgangspunkt ist der, dass Sie bereits einen OpenLDAP installiert haben, da die benötigten Kommandos Bestandteil der Pakete sind. Der Server muss für die Umwandlung noch nicht konfiguriert sein und auch nicht laufen.

Kopieren Sie als Erstes Ihr Schema, zum Beispiel company.schema, nach /opt/symas/etc/openldap/schema/. Erzeugen Sie ein leeres Unterverzeichnis /root/schema. In diesem leeren Verzeichnis wird die umgewandelte Konfiguration abgelegt. Erstellen Sie eine Datei /root/schema-ldif.conf und tragen dort die Zeile aus Listing 2.4 ein:

Listing 2.4 Konfigurationsdatei zur Umwandlung

```
include /opt/symas/etc/openldap/schema/company.schema
```

Mit dem Kommando `slaptest` können Sie jetzt die Umwandlung durchführen. Sehen Sie dazu auch das Listing 2.5:

Listing 2.5 Umwandlung mit slaptest

```
root@provider01:~# slaptest -f schema-ldif.conf -F /root/schema
config file testing succeeded
```

Wichtig: Das Verzeichnis, in dem die Dateien bei der Umwandlung abgelegt werden, muss unbedingt leer sein, sonst kommt es bei der Umwandlung zu einer Fehlermeldung, und der Vorgang wird abgebrochen.

Einspielen des Schemas

An dieser Stelle möchten wir Ihnen dann auch zeigen, wie Sie das neue Schema in den LDAP einbinden können. Bis zu diesem Punkt haben wir zwar noch keine Kommandos direkt erklärt und auch zu dem Aufbau der LDAP-Konfiguration noch nichts geschrieben, aber dieses ist die richtige Stelle, an der wir Ihnen den Vorgang erklären wollen.

Wenn Sie jetzt in das vorher angelegte Verzeichnis wechseln, finden Sie dort einen Eintrag cn=config/cn=schema/cn=0company.ldif. In der LDIF-Datei befindet sich das umgewandelte Schema. Nur können Sie das Schema so nicht einbinden, ein bisschen Handarbeit ist noch nötig.

Öffnen Sie die Datei mit einem Editor. Am Anfang der Datei entfernen Sie die ersten beiden mit einer # beginnenden Zeilen. Anschließend passen Sie die nächsten Zeilen wie in Listing 2.6 an:

Listing 2.6 Änderung am Anfang der Datei

```
<Original>
dn: cn={0}company
objectClass: olcSchemaConfig
cn: {0}company

<Änderung>
dn: cn=company,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: company
```

Am Ende der Datei entfernen Sie die Zeilen aus Listing 2.7:

Listing 2.7 Änderungen am Ende der Datei

```
structuralObjectClass: olcSchemaConfig
entryUUID: 21b3d23a-460e-103d-8c56-6f21b1f0756f
creatorsName: cn=config
createTimestamp: 20230221083316Z
entryCSN: 20230221083316.323337Z#000000#000#000000
modifiersName: cn=config
modifyTimestamp: 20230221083316Z
```

Kopieren Sie die Datei nach /opt/symas/etc/openldap/schema/company.ldif. Im Anschluss können Sie Ihre Konfiguration mit dem Kommando `ldapadd -Y EXTERNAL -H ldapi:/// -f /opt/symas/etc/openldap/schema/company.ldif` um das Schema erweitern.

Erweiterung des Schemas

Wollen Sie später weitere Attribute zu Ihrem Schema hinzufügen, können Sie das im laufenden Betrieb des LDAP-Servers realisieren. Erstellen Sie eine LDIF-Datei mit dem neuen Attribut und der geänderten Objektklasse so wie das Beispiel in Listing 2.8:

Listing 2.8 Erweiterung des Schemas

```
dn: cn={4}my-schema,cn=schema,cn=config
changetype: modify
add: olcAttributeTypes
olcAttributeTypes: {10}(1.3.6.1.4.1.12345.1.1.11
  NAME 'stka-LuckyCustomer'
  DESC 'If set to true then the customer is lucky'
  EQUALITY booleanMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE)
-
delete: olcObjectClasses
olcObjectClasses: {0}
-
add: olcObjectClasses
olcObjectClasses: {0}( 1.3.6.1.4.1.12345.1.2.1 NAME 'stka-Customer'
  DESC 'Objectclass to manage all customers'
  SUP top STRUCTURAL
  MUST ( stka-CustomerName $ stka-CustomerCity )
  MAY ( stka-CustomerPhoto $ stka-CustomerStreetName $ stka-CustomerZipCode
  $ stka-CustomerCity $ stka-CustomerCountry $ stka-CustomerPhonenumber
  $ stka-CustomerHomepage $ stka-CustomerMail $ stka-GoodCustomer $ stka-
       LuckyCustomer)
```

Wichtig: Die Nummer des Schemas in Ihrer Datei muss nicht mit dem DN aus dem Beispiel übereinstimmen. Schauen Sie in das Verzeichnis /opt/symas/etc/openldap/slapd.d/cn=config/cn=schema. Dort finden Sie alle Schemata mit der entsprechenden Nummer.

Anschließen können Sie die Änderung mit `ldapmodify -Y EXTERNAL -H ldapi:/// -f mod-company.ldif` einspielen.

Überlegen Sie genau, welche Attribute Sie in einem eigenen Schema benötigen, denn das Entfernen eines Schemas ist nicht so trivial wie das Einbinden. Denn es dürfen keine Objekte mit Attributen aus dem Schema mehr im LDAP vorhanden sein. Entfernen Sie ein Schema und es sind noch Objekte mit Attributen aus dem Schema vorhanden, ist Ihre Datenbank nicht mehr konsistent, und der LDAP-Server startet nicht mehr. Wenn Sie alle Attribute und Objektklassen entfernt haben, stoppen Sie den LDAP-Server und löschen die Schemadatei aus dem Verzeichnis /opt/symas/etc/openldap/slapd.d/cn=config/cn=schema. Anschließend starten Sie den LDAP wieder neu, und das Schema ist wieder entfernt.

2.1.8 Das LDIF-Format

Damit Sie nach der Konfiguration des LDAP-Servers die ersten Objekte in dem Baum erstellen können, benötigen Sie Dateien im *Lightweight Database Interchange Format* (LDIF).

In diesen Dateien werden die Objektklassen und Attribute mit ihren Werten für das zu erstellende Objekt angelegt und anschließend mit dem Kommando `ldapadd` in den Verzeichnisbaum eingetragen. Listing 2.9 zeigt zwei Beispiele für LDIF-Einträge:

Listing 2.9 Beispiel für eine LDIF-Datei

```
dn: ou=users,dc=example,dc=net
ou: users
objectClass: organizationalUnit

dn: uid=skania,ou=users,dc=example,dc=net
uid: skania
cn: Stefan
sn: Kania
userPassword:: e1NTSEF9OURIWjB4ZVVmekl5WHVhcC9lOHo5OWZvQkRrMFdqaFo=
loginShell: /bin/bash
uidNumber: 501
gidNumber: 100
homeDirectory: /home/skania
shadowMin: -1
shadowMax: 999999
shadowWarning: 7
shadowInactive: -1
shadowExpire: -1
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
```

Im ersten Teil wird eine neue Organisatorische Einheit (OU) erzeugt, in der später die Benutzer erstellt werden sollen. Als erste Zeile muss immer der *DN* des Objekts stehen. Das Objekt gehört zu den Objektklassen *top* und *organizationalUnit*. Nach dem letzten Attribut folgt eine Leerzeile. Diese Leerzeile dient als Trennung zwischen zwei Objekten und ist immer dann erforderlich, wenn mit einer LDIF-Datei mehrere Objekte erzeugt werden sollen. Wichtig ist, dass in der Leerzeile wirklich kein Zeichen steht, auch kein Tabulator oder Leerzeichen.

Bei dem zweiten Objekt handelt es sich um einen Benutzer. Hier sehen Sie, wie das Objekt aus mehreren Objektklassen zusammengesetzt wird, aber auch, dass nicht alle Attribute aus den Objektklassen Verwendung finden.

Hier fällt auf, dass bei dem Attribut *userPassword* ein doppelter Doppelpunkt nach dem Attributname steht. Hier handelt es sich dann um einen *BASE64*-kodierten Wert. Immer, wenn Sie ein Attribut mit Sonderzeichen in den LDAP schreiben, wird der Wert im LDAP BASE64-kodiert abgelegt. Das ist auch der Fall, wenn Sie am Ende einer Zeile ein Leerzeichen setzen. Bei vielen Einträgen, vor allen Dingen später in der dynamischen Konfiguration, kann das auch zu Problemen führen. Wenn Sie einen Wert, der in BASE64 gespeichert ist, umwandeln wollen, können Sie das mit dem Kommando `echo "BASE64-Wert" | base64 -d` realisieren.

Tipp: Wenn Sie den *vi* für die Bearbeitung der LDIF-Dateien nutzen, können Sie im Kommando-Modus *: set list* eingeben, dann werden Ihnen alle Leerzeichen und Tab-Sprünge angezeigt. So können Sie dann die Leerzeichen am Ende einer Zeile einfach finden und entfernen.

Ein weiterer Punkt soll hier noch erwähnt werden: Laut RFC 2849 ist die maximale Zeilenlänge für einen im LDAP gespeicherten Eintrag 76 Zeichen, längere Zeilen werden um-

brochen. Eine umbrochene Zeile können Sie an genau einem führenden Leerzeichen erkennen. In einem Editor würde diese Zeile aber als neue Zeile interpretiert. Wenn Sie sich Objekte aus dem LDAP auflisten lassen, die längere Zeilen haben, können Sie die Ausgabe durch die Kommandokette `echo "Langezeile" | perl -p0e "s/\n //g"` wieder zu einer Zeile zusammenfügen. Sie können auch die gesamte Ausgabe von `ldapsearch` durch die Weiterleitung ausgeben lassen. Alle Zeilen mit Umbrüchen werden dann zusammenhängend ausgegeben.

Mithilfe der LDIF-Dateien können Sie beliebig viele Objekte auf einmal erstellen oder verändern. Zusammen mit einem Shell-Skript wäre es auch möglich, die entsprechenden Werte für die Attribute aus einer anderen Datei auszulesen und damit komplexe Änderungen an vielen Objekten innerhalb des gesamten DITs durchzuführen.

2.1.9 Aufbau einer Struktur

Immer wieder werden wir gefragt: „Was ist denn nun die beste Struktur für einen Verzeichnisdienst?“ Leider kann niemand diese Frage universell beantworten, sondern das ist immer abhängig vom Einzelfall. Aber die Zusammensetzung eines Verzeichnisbaums kann erklärt werden.

Ein Verzeichnisbaum setzt sich grundlegend aus zwei verschiedenen Objektarten zusammen. Da sind zum einen die *Organizational Units* (OU), die eine Struktur aufbauen, und zum anderen die *Blattobjekte*, die die Ressourcen beschreiben und bereitstellen. Bei OUs gibt es jetzt verschiedene Möglichkeiten, wie Sie die Struktur aufbauen können. Die oberste Ebene eines Verzeichnisbaums ist immer der *root Directory Service Entry* (rootDSE). Diese Ebene ist eine unsichtbare Ebene, in der bestimmte Informationen über die Fähigkeiten in den *optional attributes* festgelegt werden. Erst unterhalb des rootDSE werden Sie Ihren eigenen Teil des Verzeichnisbaums erstellen. Die erste Ebene, die Sie dort erstellen, ist immer Ihr *BaseDN*, wobei *DN* für *Distinguished Name* steht, den eindeutigen Namen Ihres Verzeichnisdienstes. In Tabelle 2.2 sehen Sie eine Übersicht über die verschiedenen OU-Typen, die Sie verwenden können:

Tabelle 2.2 Organizational-Unit-Objekte

Abkürzung	Name	Beschreibung	Beispiel
C	Country	2-stelliger Ländercode	C=DE
O	Organization	Organisationsname	O=Firma
DC	domainComponent	DNS-Name	DC=firma,DC=de
OU	organizationalUnitName	Name für organisatorische Einheiten	OU=Abteilungen

Sie haben damit die Möglichkeit, eine Struktur Ihres Unternehmens so abzubilden, dass die Struktur die Gliederung Ihres Unternehmens weltweit widerspiegelt, oder Sie können den LDAP-Verzeichnisbaum an die DNS-Struktur Ihres Unternehmens anpassen. In den meisten Fällen ist es sinnvoll, die Strukturen der beiden Verzeichnisdienste LDAP und DNS anzugleichen. Genau das macht Microsoft mit seinem Active Directory. Um Ihnen beide Möglichkeiten zu veranschaulichen, sehen Sie in Bild 2.2 eine Struktur, die mit einer Länderkennung beginnt.

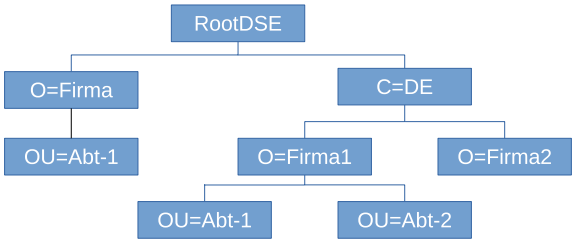

Bild 2.2 Struktur mit Länderkennung

In Bild 2.3 sehen Sie die Gliederung über das DC-Objekt. So können Sie dann Ihre Struktur passend zum DNS-Verzeichnisdienst aufbauen.

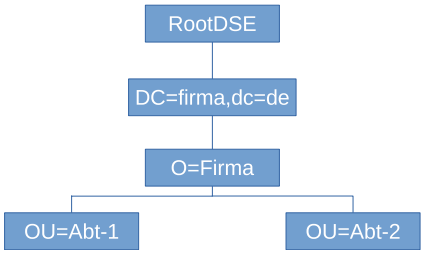

Bild 2.3 Struktur mit einem DC-Objekt

Die weitere Planung über die untergeordneten OUs sollten Sie sehr genau vornehmen. Eine falsche oder schlechte Planung sorgt später dafür, dass es Ihnen schwerfallen kann, die Berechtigungen in Ihrem Verzeichnisdienst zu setzen.

2.1.10 Namensfindung

Damit ein Objekt im Verzeichnisbaum auch eindeutig adressiert werden kann, müssen Sie wissen, wie Sie ein Objekt eindeutig ansprechen können. Jedes Objekt im LDAP hat einen eindeutigen Namen, den *Distinguished Name* (DN). Dieser Name wird immer dann benötigt, wenn Sie ein neues Objekt anlegen oder ein bestehendes Objekt verändern wollen. Als Beispiel sehen Sie hier den DN der OU *OU=Abt-1* aus Bild 2.3 *ou=Abt-1,O=Firma,DC=firma,dc=de*. Wie Sie an dem Namen sehen, wird der Name immer vom eindeutigen zum abstrakten Element geschrieben, genau wie im DNS, bei dem es sich auch um einen Verzeichnisdienst handelt.

Neben dem *DN* gibt es noch den *RDN*, den *Relative Distinguished Name*. Dabei handelt es sich immer um den Namen, den ein Objekt relativ zum eigenen Standpunkt im Verzeichnisdienst hat. Sie können den RDN gut mit dem relativen Dateinamen im Dateisystem vergleichen; dieser Name ist auch immer abhängig von meinem aktuellen Standort im Dateisystem.

3 Installation des ersten OpenLDAP

In der ersten Auflage dieses Buches haben wir die Installation noch für zwei Distributionen beschrieben, da bei den verschiedenen Distributionen ein paar Unterschiede bestanden. Dieses Mal greifen wir auf die Pakete der Firma Symas zurück, daher sind jetzt, bei allen unterstützten Distributionen, die Pfade und Dateinamen identisch, sodass wir hier die Installation distributionsunabhängig beschreiben können.

Hinweis: Bei den Paketen ist der Pfad, in dem die Dateien der Pakete abgelegt werden, immer /opt/symas/. Der Grund dafür ist: So kollidieren die Dateien aus den Symas-Paketen nicht mit den Dateien aus den Paketen der Distribution, falls Sie die Pakete noch installiert haben.

An dieser Stelle gehen wir auch noch einmal auf die Unterschiede zwischen der statischen und dynamischen Konfiguration ein. Im Rest des Buches werden wir dann ausschließlich die dynamische Konfiguration nutzen.

Der Grund ist der, dass die statische Konfiguration auf der Website *https://www.openldap.org* als *deprecated* angegeben wird. Das bedeutet, dass die Möglichkeit der Konfiguration über eine einfache ASCII-Textdatei leider auf Dauer nicht mehr möglich sein wird. In komplexen Installationen mit mehreren Providern ist die Konfiguration über die statische Konfiguration auch nicht mehr zeitgemäß.

Bei der dynamischen Konfiguration werden sowohl die Grundkonfiguration als auch alle weiteren Änderungen über LDIF-Dateien in einer speziellen Datenbank im LDAP gespeichert, und der LDAP-Server verwaltet sich quasi selbst.

3.0.1 Die statische Konfiguration

Bei der statischen Konfiguration handelt es sich um die klassische Variante der Konfiguration. Alle Konfigurationsparameter werden in einer Konfigurationsdatei abgelegt. Immer, wenn Sie eine Änderung durchgeführt haben, ist es notwendig, dass Sie den LDAP-Server neu starten, damit die Änderung wirksam wird. Wenn Sie mehrere LDAP-Server einsetzen, müssen Sie die Konfiguration immer auf allen LDAP-Servern anpassen.

Nicht nur die grundlegende Konfiguration wird in der statischen Konfiguration in der Konfigurationsdatei abgelegt, sondern später auch alle ACLs und die Konfiguration der eventu-

ell verwendeten Overlays (bei Overlays handelt es sich um eine Art Plug-in, das die Funktion des LDAP-Servers erweitert; mehr zum Thema Overlays finden Sie in Kapitel 10, «Einsatz von Overlays»).

Sie können die Konfiguration Ihres LDAP-Servers auch erst statisch beginnen und später auf die dynamische Konfiguration umstellen.

Da die Datei slapd.conf bei allen Distributionen denselben Inhalt hat, möchten wir Ihnen an dieser Stelle die einzelnen Zeilen der Konfiguration erklären. So können Sie in diesem Kapitel den Inhalt der statischen Konfiguration noch sehr gut mit der dynamischen Konfiguration vergleichen. In Listing 3.1 sehen Sie den Inhalt der Datei:

Listing 3.1 Die slapd.conf

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /opt/symas/etc/openldap/schema/core.schema
include         /opt/symas/etc/openldap/schema/cosine.schema
include         /opt/symas/etc/openldap/schema/nis.schema
include         /opt/symas/etc/openldap/schema/inetorgperson.schema

pidfile         /var/symas/run/slapd.pid
argsfile        /var/symas/run/slapd.args

# Read slapd.conf(5) for possible values
loglevel        256

# Load dynamic backend modules:
modulepath      /opt/symas/lib/openldap
moduleload      back_mdb.la
moduleload      argon2.la

# The maximum number of entries that is returned for a search operation
sizelimit 500

# The tool-threads parameter sets the actual amount of cpu's that is used
# for indexing.
tool-threads 1
password-hash {ARGON2}

########################################
# Specific Directives for database #1, of type hdb:
# Database specific directives apply to this databasse until another
# 'database' directive occurs
database        mdb
maxsize         1073741824
# The base of your directory in database #1
suffix          "dc=example,dc=net"

# rootdn directive for specifying a superuser on the database.
rootdn          "cn=admin,dc=example,dc=net"
#rootpw          "geheim"
rootpw          "{ARGON2}$argon2i$v=19$m=4096,t=3,p=1$c2Fs..."
```

```
# Where the database file are physically stored for database #1
directory       "/var/symas/openldap-data"

# Indexing options for database #1
index           objectClass eq

# Save the time that the entry gets modified, for database #1
lastmod         on


# The userPassword by default can be changed
# by the entry owning it if they are authenticated.
# Others should not be able to see it, except the
# admin entry below
# These access lines apply to database #1 only

access to attrs=userPassword,shadowLastChange
        by anonymous auth
        by self write
        by * none

access to *
        by * read

access to dn.base="" by * read
```

Die Konfigurationsdatei besteht aus zwei Teilen: Der erste Teil oberhalb der Linie aus Hashes ist der globale Teil, der für alle Datenbanken, die der OpenLDAP bereitstellt, relevant ist. Der Teil unterhalb betrifft immer eine bestimmte Datenbank. Der Parameter *database mdb* markiert dabei den Beginn einer neuen Datenbank.

- Als Erstes werden die Schematadateien eingebunden, die Reihenfolge ist dabei sehr wichtig. Beim Start des LDAP-Servers werden die Dateien eine nach der anderen abgearbeitet. Da durch die Möglichkeit der Vererbung von Attributen eine Abhängigkeit zwischen den einzelnen Schemata besteht, müssen Sie immer wissen, welche Objektklasse eventuell Attribute einer anderen Objektklasse nutzt. Die Standardobjektklassen, die hier eingebunden sind, befinden sich schon in der richtigen Reihenfolge. Wenn Sie ein eigenes Schema mit eigenen Objektklassen erstellen, ist es wichtig, dass Sie am Anfang Ihres Schemas etwaige Abhängigkeiten kommentieren.
- In den Zeilen *pidfile /var/symas/run/slapd.pid* und *argsfile /var/symas/run/slapd.args* werden die Prozess-ID und die Argumente, die beim Start des OpenLDAP-Servers verwendet werden, abgelegt.
- Das *loglevel* legt fest, welche Ereignisse geloggt werden. Dabei werden die Zahlen nicht einfach hochgezählt, sondern die einzelnen Werte werden binär übergeben. Am Anfang ist ein *loglevel 256 (stats)* eine gute Einstellung, denn da werden alle Zugriffe geloggt. Welche Loglevel Sie verwenden können, finden Sie in der Manpage zur slapd.conf. Im späteren produktiven Betrieb stellt jede Art von Logging immer einen Flaschenhals da. Es ist daher sinnvoll, nachdem der OpenLDAP eingerichtet wurde, das Loglevel auf den Wert *none* zu setzen, dann werden nur noch systemkritische Fehler ins Log geschrie-

ben. Setzen Sie das Loglevel auf *0*, werden auch diese Meldungen nicht mehr ins Log geschrieben.

- Die Parameter *modulepath* und *moduleload* werden immer dann benötigt, wenn Sie zusätzliche Module für zusätzliche Funktionen nutzen wollen. Bei den Symas-Paketen werden alle Overlays über Module bereitgestellt und müssen einzeln geladen werden.
- Der Parameter *sizelimit 500* legt fest, wie viele Antworten bei einer Suche zurückgegeben werden. Gerade wenn Sie einen großen LDAP-Baum mit mehreren Tausend Objekten pflegen, kann die Antwort den Server stark belasten. Deshalb diese Grenze. Über Filter können Sie die Suche einschränken. Die Übersteuerung der Einschränkung der zurückgegebenen Objekte bei der Suche muss immer direkt in der Datenbank definiert werden, für die sie zutreffen soll.
- Wenn Sie sehr viele Indexdatenbanken einsetzen, um die Suche nach Attributen oder Objekten zu beschleunigen, kann es sinnvoll sein, mehr als nur eine CPU für das Indizieren zu verwenden. Testen Sie, ob eine weitere CPU für Sie Vorteile bringt.
- Wir werden hier im Buch nur noch *ARGON2* als Passworthash nutzen. Über den Parameter *password-hash* legen Sie den neuen Passworthash fest. Mehr zum Thema *ARGON2* finden Sie unter *https://de.wikipedia.org/wiki/Argon2*.
- Mit *database mdb* beginnt nicht nur der Datenbankteil, sondern Sie legen auch den Datenbanktyp fest. Seit einiger Zeit wird nur noch der Datenbanktyp *mdb* empfohlen, alle anderen alten Datenbanktypen werden in Zukunft nicht mehr unterstützt.
- Der *suffix* legt die oberste Ebene Ihres LDAP-Baums fest.
- Beim *rootDN* handelt es sich um den Hauptadministrator, der immer alle Rechte hat und nie auf irgendeine Art und Weise beschränkt werden kann. Der Benutzer wird nicht in der Datenbank angelegt.
- Das *rootpw* ist das Passwort für den *rootDN*. Halten Sie dieses Passwort immer unter Verschluss. Mit diesem Passwort können sämtliche Änderungen am LDAP-Baum vorgenommen werden. In der Beispieldatei sehen Sie, dass hier schon *ARGON2* als Passorthash genutzt wird. Anders als bei der Verwendung anderer Passwordhashes nutzen Sie hier das Kommando `echo -n "mein-geheimes-pw" | argon2 "saltsaltsaltsalt" -e`, um den Passworthash zu erzeugen.
- Mit dem Parameter *directory* legen Sie fest, in welchem Verzeichnis die Datenbankdateien abgelegt werden. Für jede Datenbank, die Sie mit dem LDAP-Server verwalten wollen, benötigen Sie ein eigenes Verzeichnis.
- Über den Parameter *index* werden die Indexdatenbanken für diese LDAP-Datenbank festgelegt. Später werden wir noch näher auf die Indexeinträge eingehen.
- *Lastmod on* speichert die Zeit der letzten Änderung für jedes Objekt in der Datenbank.
- Am Schluss folgen die ACLs für die Zugriffe. Auf die ACLs gehen wir in Kapitel 9, «Berechtigungen mit ACLs», näher ein.

Tipp: Haben Sie bis jetzt Ihren LDAP-Server über die statische Konfiguration verwaltet, können Sie diese mit dem Kommando `slaptest -F /opt/symas/etc/openldap/slapd.d -f /etc/ldap/slapd.conf` in die dynamische Konfiguration umwandeln.

3.0.2 Die dynamische Konfiguration

Bei der dynamischen Konfiguration werden alle Einstellungen des LDAP-Servers in einer eigenen Datenbank im LDAP abgelegt: Sowohl die Grundkonfiguration, die Erweiterung des Funktionsumfang durch Overlays als auch die ACLs werden mittels LDIF-Dateien in die Datenbank eingespielt. Kommentare können Sie in dieser Datenbank nicht ablegen, sodass Sie sämtliche Änderungen und Einstellungen extern dokumentieren müssen.

Der große Vorteil der dynamischen Konfiguration ist, dass Sie hier den LDAP-Server nach einer Änderung nicht neu starten müssen; die Änderung ist sofort wirksam, nachdem Sie die LDIF-Datei eingespielt haben. Auch Ihre ACLs passen Sie dann immer über LDIF-Dateien an; auch sie sind nach dem Einspielen sofort wirksam. Wenn Sie mit grafischen Werkzeugen arbeiten wollen, dann achten Sie darauf, ob das Werkzeug die dynamische Konfiguration bearbeiten kann. Wir werden in Kapitel 7, «Grafische Werkzeuge», noch darauf zu sprechen kommen.

Wenn Sie mehrere LDAP-Server einsetzen und diese vielleicht auch noch an unterschiedlichen Standorten stehen und Sie häufig Änderungen an der Konfiguration oder den ACLs vornehmen, dann ist die dynamische Konfiguration auf jeden Fall der bessere Weg. Mit der neuen Version 2.5 und 2.6 funktioniert jetzt auch die Replikation der dynamischen Konfiguration fehlerfrei. So können Sie Änderungen an der Konfiguration auf einem Server einspielen, und die Änderung wird auf alle anderen OpenLDAP-Servern repliziert. Durch den geänderten Replikationsprozess in den neuen Versionen können Sie so einen Multiprovider-Cluster aufbauen und die Konfiguration für den gesamten Cluster mit einer LDIF-Datei anpassen.

3.1 Installation der Symas-Pakete

Eine Besonderheit der Symas-Pakete sei hier gleich am Anfang erwähnt: Der Dienst startet in der Standardeinstellung, nach der Installation, grundsätzlich mit der Benutzerkennung *root*. Das ist aber keine gute Idee und soll hier auch sofort geändert werden.

Aber als Erstes werden jetzt die Pakete installiert. Dafür benötigen Sie die Daten, um das Repository auf Ihrem System einzutragen. Damit Sie nach dem Installieren der Pakete auch gleich die Zugriffsrechte für die verwendeten Verzeichnisse anpassen, sehen Sie in Listing 3.2 ein Skript, mit dem Sie nicht nur die Pakete installieren können, sondern auch gleich einen Benutzer für den Dienst anlegen und die Berechtigungen im Dateisystem setzen:

Listing 3.2 Skript zur Installation der Symas-Pakete

```
#!/bin/bash
DEBIAN_FRONTEND=noninteractive apt install -yq gnupg2 argon2 python3-ldap
wget -O- https://repo.symas.com/repo/gpg/RPM-GPG-KEY-symas-com-signing-key\
     | gpg --dearmor | tee /etc/apt/trusted.gpg.d/symas-com-gpg.gpg \
     > /dev/null
echo "deb [arch=amd64] https://repo.symas.com/repo/deb/main/release26 \
     bullseye main" | tee -a /etc/apt/sources.list.d/symas26.list
```

```
apt update -y
groupadd -r  openldap
useradd -r -g openldap -d /opt/symas/ openldap
DEBIAN_FRONTEND=noninteractive apt install -yq symas-openldap-clients \
     symas-openldap-server
chown openldap:openldap /var/symas/openldap-data/
chmod 770 /var/symas/openldap-data/
chown openldap:openldap /var/symas/run
chmod 770 /var/symas/run
mkdir -m 770 /opt/symas/etc/openldap/slapd.d
chown openldap:openldap /opt/symas/etc/openldap/slapd.d
```

Nachdem die Pakete installiert sind, benötigen Sie jetzt noch ein neues Skript für den *systemd*, um auch den *slapd* mit dem gerade angelegten Benutzer und der Gruppe zu starten. Das Skript aus Listing 3.3 können Sie so übernehmen:

Listing 3.3 Serviceskript zum Starten des slapd

```
[Unit]
Description=Symas OpenLDAP Server Daemon
After=network-online.target
Documentation=man:slapd
Documentation=man:slapd-config
Documentation=man:slapd-mdb

[Service]
Type=notify
EnvironmentFile=/etc/default/symas-openldap
ExecStart=/opt/symas/lib/slapd -d 0 -h ${SLAPD_URLS}  \
          -u ${SLAPD_USER} -g ${SLAPD_GROUP} \
          ${STAT_DYN} ${SLAPD_CONF}
[Install]
WantedBy=multi-user.target
```

Auf einem Debian-System kopieren Sie das Service-Skript in das Verzeichnis /lib/systemd/system/, auf einem Redhat-System in das Verzeichnis /usr/lib/systemd/system/.

Die hier verwendeten Variablen werden in dem Skript /etc/default/symas-openldap definiert. Wenn Sie eine andere Distribution als Debian nutzen, legen Sie das Verzeichnis /etc/default an oder ändern Sie den Pfad passend zu Ihrer Distribution ab.

Fehlt nur noch die Datei symas-openldap mit den vordefinierten Variablen. Den Inhalt der Datei sehen Sie in Listing 3.4:

Listing 3.4 Konfiguration der Variablen für den Start

```
SLAPD_URLS="ldap:/// ldapi:/// ldaps:///"
SLAPD_OPTIONS=""
SLAPD_USER="openldap"
SLAPD_GROUP="openldap"
# Select static "-f" or dynamic "-F"
STAT_DYN="-F"
# To use static configuration
#SLAPD_CONF="/opt/symas/etc/openldap/slapd.conf"
# To use dynamic configuration
SLAPD_CONF="/opt/symas/etc/openldap/slapd.d"
```

Hier wird gleich die dynamische Konfiguration eingestellt, und es werden alle drei möglichen Protokolle für den Zugriff auf den LDAP-Server aktiviert. Auch sehen Sie hier, dass der Benutzer und die Gruppe für den Start des Dienstes festgelegt wird.

Die dynamische Konfiguration

Nachdem Sie im vorigen Abschnitt gesehen haben, wie die Konfiguration über die Datei slapd.conf durchgeführt wird, zeigen wir Ihnen in diesem Abschnitt die dynamische Konfiguration.

Die dynamische Konfiguration wird in Form von LDIF-Dateien im Verzeichnis /opt/symas/etc/openldap/slapd.d abgelegt. In dem Verzeichnis finden Sie, nachdem Sie die Konfiguration eingespielt haben, weitere Unterverzeichnisse und LDIF-Dateien, denn die Konfiguration wird nicht in einem großen LDIF-File abgelegt, sondern setzt sich aus mehreren LDIF-Dateien zusammen. Die LDIF-Dateien sind dort nach Datenbanken und Inhalten in unterschiedlichen Verzeichnissen abgelegt. Beim Neustart des LDAP-Servers wird die Konfiguration aus den LDIF-Dateien geladen.

Jetzt stehen wir aber erst einmal vor dem Henne-Ei-Problem. Der *slapd* startet nicht ohne Konfiguration, und die Konfiguration wird über die ldap-Kommandos in den LDAP geschrieben. Aber der Dienst läuft nicht, weil keine Konfiguration da ist.

Aber zum Glück lässt sich das Problem recht einfach lösen. Mithilfe des Kommandos `slapadd` können Sie Daten in bestehende Datenbanken schreiben oder neue Datenbanken erstellen, ohne dass der *slapd* läuft. Sie benötigen lediglich eine LDIF-Datei, in der die Grundkonfiguration definiert ist, und können diese dann mit `slapadd` einspielen.

Warum klappt das? Die `slap`-Kommandos greifen direkt auf die Datenbankdateien zu und gehen nicht den Weg über das LDAP-Frontend *slapd*. Aus diesem Grund ist es auch wichtig, dass bei der Verwendung von `slapadd` der Dienst nicht läuft, da Sie sonst direkt in geöffnete Datenbankdateien schreiben würden. Daher können wir auch auf diesem Weg die Konfiguration in den OpenLDAP einspielen. Doch bevor wir die Konfiguration einspielen, werfen wir erst einmal einen Blick auf eine Konfigurationsdatei für die Grundkonfiguration und schauen uns einmal an, wo wir, parallel zur statischen Konfiguration, die Informationen finden. Listing 3.5 zeigt die Konfigurationsdatei im LDIF-Format:

Listing 3.5 LDIF-Datei mit den Grundeinstellungen

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcLogLevel: sync
olcLogLevel: stats
olcPidFile: /var/symas/run/slapd.pid
olcArgsFile: /var/symas/run/slapd.args
olcToolThreads: 1

dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema

dn: cn=module{0},cn=config
```

```
objectClass: olcModuleList
cn: module{0}
olcModulePath: /opt/symas/lib/openldap
olcModuleLoad: back_mdb
olcModuleLoad: back_monitor
olcModuleLoad: argon2.la

include: file:///opt/symas/etc/openldap/schema/core.ldif
include: file:///opt/symas/etc/openldap/schema/cosine.ldif
include: file:///opt/symas/etc/openldap/schema/nis.ldif
include: file:///opt/symas/etc/openldap/schema/inetorgperson.ldif

dn: olcDatabase={-1}frontend,cn=config
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: {-1}frontend
olcSizeLimit: 500
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
  by * break
olcAccess: {1}to dn=""  by * read
olcAccess: {2}to dn.base="cn=subschema"  by * read
olcPasswordHash: {ARGON2}

dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcRootDN: cn=admin,cn=config
olcRootPW: {ARGON2}$argon2i$v=19$m=4096,t=3,p=1$cX....
olcAccess: {0}to *
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage

dn: olcDatabase={1}monitor,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {1}monitor
olcAccess: {0}to dn.subtree="cn=monitor"
  by dn.exact=cn=admin,cn=config read
  by dn.exact=cn=admin,dc=example,dc=net read
  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth read

dn: olcDatabase={2}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {2}mdb
olcSuffix: dc=example,dc=net
olcRootDN: cn=admin,dc=example,dc=net
olcRootPW: {ARGON2}$argon2i$v=19$m=4096,t=3,p=1$cM....
olcDbCheckpoint: 512 30
olcDbDirectory: /var/symas/openldap-data
olcDbIndex: default eq
olcDbIndex: objectClass
olcDbMaxSize: 85899345920
olcAccess: {0} to attrs=userPassword
  by anonymous auth by self write by * none
olcAccess: {1} to attrs=shadowLastChange
```

```
    by anonymous auth by self write by * none
olcAccess: {2} to *
    by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage
    by * read
```

Wenn Sie diese Einträge mit der statischen Konfiguration vergleichen, finden Sie hier alle Einstellungen wieder, nur werden jetzt die Informationen aus einer eigenen Datenbank gelesen. Jeder Teilbereich der Konfiguration wird in einem eigenen *DN* beschrieben, die einzelnen Bereiche werden für die folgenden Einstellungen benutzt:

- *dn: cn=config*
  Die Grundkonfiguration der globalen Konfiguration, auch in der statischen Konfiguration finden Sie diese Einträge im globalen Bereich wieder.
- *dn: cn=schema,cn=config*
  Auch die von Ihnen konfigurierten Schemata werden in der Konfigurationsdatenbank abgelegt und erhalten hierfür einen eigenen Bereich in Form eines *DN*.
- *dn: cn=module{0},cn=config*
  Hier handelt es sich um den Teil der Konfiguration, in dem alle benötigten Module eingetragen werden. Sie sehen hier, dass dort auch gleich das Modul für den Passwordhash *ARGON2* eingetragen wird, damit der Hash auch schon für die Passwörter der *rootdn* verwendet werden kann. Im Verlauf des Buchs werden wir die Liste kontinuierlich erweitern.
- *dn: olcDatabase={-1}frontend,cn=config*
  Das ist der zweite Teil der globalen Konfiguration; hier können Sie schon ACLs festlegen, wer diesen Bereich verwalten darf. Auch werden hier ACLs eingetragen, die für alle Datenbanken auf dem LDAP-Server gelten sollen. Der Zugriff auf die oberste Ebene und auf die Schemata muss immer gewährleistet sein. Durch den Eintrag an dieser Stelle muss der Zugriff nicht immer für jede Datenbank einzeln vergeben werden.
- *dn: olcDatabase={1}monitor,cn=config*
  Hier sehen Sie die erste Datenbank, die nicht an eine feste Reihenfolge gebunden ist. Alle vorherigen *DNs* haben grundsätzlich immer dieselbe Nummer. Ab jetzt hängt die Nummerierung der Datenbanken immer davon ab, in welcher Reihenfolge Sie die Datenbank definieren. Deshalb ist es ganz wichtig, dass Sie ab jetzt bei jeder Änderung der Konfiguration genau auf die Nummerierung der Datenbanken achten.

  Bei der Datenbank, die hier definiert wird, handelt es sich um die Datenbank, in der alle Aktionen, die ein Client auf dem LDAP-Server ausführt, protokolliert werden. Diese Daten können Sie in Ihrem Monitoring auswerten, und so haben Sie immer einen guten Überblick über den Zustand Ihres LDAP-Servers.
- *dn: olcDatabase={2}mdb,cn=config*
  Bei diesem *DN* handelt es sich jetzt um die Konfiguration der Objektdatenbank. Auch hier finden Sie alle Parameter aus der statischen Konfiguration wieder.

Eine Sache fällt sofort ins Auge: Die Parameter haben hier andere Namen und beginnen immer mit *olc*. Die Abkürzung steht für *Open LDAP Configuration*. Die Schreibweise der Parameter ist etwas anders; wenn Sie einen bestimmten Parameter suchen, finden Sie die Schreibweise immer in den LDIF-Dateien im Verzeichnis /opt/symas/etc/openldap/schema. Später können Sie auch in der dynamischen Konfiguration nach den Namen suchen, dazu muss der LDAP-Server aber erst laufen.

Hinweis: Wenn Sie bereits Erfahrung mit der dynamischen Konfiguration im OpenLDAP 2.4 gesammelt haben und dort auch schon andere Passworthashes nutzen, gibt es ab der Version 2.5 eine Änderung. Der Parameter *olcPasswordHash* befindet sich jetzt im DN *dn: olcDatabase={-1}frontend,cn=config* und nicht mehr im DN *dn: cn=config*.

Einen ganz wichtigen Punkt haben wir bis jetzt noch nicht angesprochen. In fast jedem Bereich der Konfiguration finden Sie die folgende ACL *by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth manage*. Was hat es damit auf sich? Diese ACL kann Ihnen das Leben, gerade am Anfang, etwas vereinfachen. Schauen wir uns die einzelnen Teile der ACL mal etwas genauer an:

- *dn.exact*
  Der folgende *DN* muss exakt mit dem hier angegebenen Wert übereinstimmen.
- *gidNumber=0+uidNumber=0*
  Die ACL trifft also auf den Benutzer mit der UID und GID *0* zu, also auf den *root*.
- *cn=peercred,cn=external,cn=auth*
  Hier wird eine im OpenLDAP vorkonfigurierte SASL-Authentifizierung verwendet. Diese Art der Authentifizierung ist nur lokal auf dem Server über den lokalen Socket *ldapi* möglich.
- *manage*
  Der Benutzer erhält den vollen Zugriff auf alle Objekte und Einträge.

Das heißt also, der Benutzer *root* hat direkt auf dem LDAP-Server vollen Zugriff auf alle Einträge, ohne sich zusätzlich mit einem Passwort authentifizieren zu müssen.

Hinweis: Diese Grundkonfiguration können Sie für alle folgenden LDAP-Server als Grundlage nehmen, denn hier sind noch keine rollenspezifischen Einstellungen enthalten.

Nachdem die Grundkonfiguration des OpenLDAP-Servers eingespielt ist, werden wir diese Art der Authentifizierung anhand des Kommandos `ldapsearch` erklären.

Einspielen der dynamischen Konfiguration

Kommen wir jetzt zur Lösung des Henne-Ei-Problems: das Einspielen der Konfiguration in die Datenbank. Wie vorher schon erwähnt, wird dafür das Kommando `slapadd` verwendet. Mithilfe des Kommandos `slapadd` können Sie Daten in verschiedenen Datenbanken des OpenLDAP einlesen. Daher benötigt das Kommando immer eine Zieldatenbank und eine Quelldatei, aus der die Informationen gelesen werden sollen. Neben diesen beiden Parametern wird ein weiterer Parameter benötigt, nämlich die Nummer der Datenbank, in die geschrieben werden soll. Da wir hier die Konfiguration schreiben wollen, ist das immer die Nummer *0*. In Listing 3.6 sehen Sie das Kommando und die Ausgabe nach erfolgreicher Einspielung der Konfiguration:

Listing 3.6 Einspielung der initialen Konfiguration

```
root@provider01:~# slapadd -n0 -F /opt/symas/etc/openldap/slapd.d/ -l config.ldif
Closing DB...
```

- *-n0*
  Hierbei handelt es sich um die Nummer der Datenbank innerhalb der Konfiguration.
- *-F /opt/symas/etc/openldap/slapd.d/*
  Das Zielverzeichnis für die Konfiguration.
- *-l config.ldif*
  Der Name (eventuell mit Pfad) der LDIF-Datei, in der sich die Konfiguration befindet.

Sollten Sie beim Einspielen der Konfiguration einen Fehler angezeigt bekommen, löschen Sie erst den Inhalt des Verzeichnisses /opt/symas/etc/openldap/slapd.d, bevor Sie einen erneuten Versuch starten.

Wichtig: Sie spielen die Konfiguration als Benutzer *root* ein, also gehören auch alle dabei entstandenen Dateien immer dem Benutzer *root*. Mit dem Kommando `chown -R openldap: /opt/symas/etc/openldap/slapd.d/` ändern Sie den Besitzer und die besitzende Gruppe auf *openldap*. Erst dann können Sie den OpenLDAP mit dem Kommando `systemctl start symas-openldap-server.service` starten.

Erster Zugriff auf die dynamische Konfiguration

Nachdem Sie die Konfiguration eingespielt haben und der *slapd* mit der neuen Konfiguration läuft, können Sie jetzt auf den LDAP zugreifen. Schauen Sie sich aber als Erstes einmal in dem Verzeichnis /opt/symas/etc/openldap/slapd.d um. Dort finden Sie die gesamte Konfiguration, unterteilt in einzelne Verzeichnisse und LDIF-Dateien. Als Beispiel soll hier einmal ein Blick in die Datei cn=config/olcDatabase={2}mdb.ldif geworfen werden. Listing 3.7 zeigt den Inhalt der Datei:

Listing 3.7 Inhalt einer Konfigurationsdatei

```
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 1bf6383c
dn: olcDatabase={2}mdb
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {2}mdb
olcDbDirectory: /var/symas/openldap-data
olcSuffix: dc=example,dc=net
olcAccess: {0} to attrs=userPassword by anonymous auth by self write \
               by * none
olcAccess: {1} to attrs=shadowLastChange by anonymous auth by self write \
               by * none
olcAccess: {2} to *  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,\
               cn=extern
 al,cn=auth manage by * read
olcRootDN: cn=admin,dc=example,dc=net
olcRootPW:: e0FSR09OMn0kYXJnb24yaSR2PTE5JG09NDA5Nix0PTMscD0xJGNYZGxjbkowZW5WNm
 RXbHZNVEl6JEcvbDBseW5mN3lnZHowdEcrRTdTMWZCaWJzRnMvTDgwQVVTaXNpR2wvdjQ=
olcDbCheckpoint: 512 30
olcDbIndex: default eq
olcDbIndex: objectClass
olcDbMaxSize: 85899345920
structuralObjectClass: olcMdbConfig
```

```
entryUUID: 238e0ba8-301e-103d-9d88-8b16491810fd
creatorsName: cn=admin,cn=config
createTimestamp: 20230124103225Z
entryCSN: 20230124103225.801378Z#000000#000#000000
modifiersName: cn=admin,cn=config
modifyTimestamp: 20230124103225Z
```

Am Anfang der Datei steht der Hinweis, dass Sie die Datei nicht mit einem Editor bearbeiten sollen. Das ist auch richtig so, denn in der zweiten Zeile wird eine *CRC32*-Prüfsumme angezeigt. Wenn Sie die Datei mit einem Editor bearbeiten, ändert sich auch die Prüfsumme. Beim Starten des LDAP-Servers werden diese Prüfsummen aber kontrolliert. Wenn eine Prüfsumme einer Datei nicht stimmt, dann startet der LDAP-Server nicht mehr und zeigt auch im Log eine entsprechende Meldung an. Im Anschluss daran finden Sie alle Einstellungen der entsprechenden Datenbank, gefolgt von den internen Attributen, die der OpenLDAP für die Verwaltung dieser Datenbank benötigt.

Wenn Sie sich jetzt den Inhalt der Konfiguration ansehen wollen, benötigen Sie dafür das Kommando `ldapsearch`. In Listing 3.8 sehen Sie das Kommando mit einer gekürzten Ausgabe des Ergebnisses. Für die Authentifizierung verwenden wir hier den SASL-Mechanismus *EXTERNAL* unter Verwendung der lokalen Schnittstelle *ldapi*. Die Zugriffsrechte erhalten Sie hier über die vorher schon angesprochene ACL, nur als *root*:

Listing 3.8 Suche nach der Konfiguration

```
root@provider01:~# ldapsearch -Y EXTERNAL -H ldapi:/// \
                   -b cn=config -LLL | less
dn: cn=config
objectClass: olcGlobal
cn: config
olcArgsFile: /var/symas/run/slapd.args
olcLogLevel: sync
olcLogLevel: stats
olcPidFile: /var/symas/run/slapd.pid
olcToolThreads: 1

dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModulePath: /opt/symas/lib/openldap
olcModuleLoad: {0}back_mdb
olcModuleLoad: {1}back_monitor
olcModuleLoad: {2}argon2.la
...
dn: olcDatabase={2}mdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcMdbConfig
olcDatabase: {2}mdb
olcDbDirectory: /var/symas/openldap-data
olcSuffix: dc=example,dc=net
olcAccess: {0} to attrs=userPassword by anonymous auth by self write \
               by * none
olcAccess: {1} to attrs=shadowLastChange by anonymous auth by self write \
               by * none
olcAccess: {2} to *  by dn.exact=gidNumber=0+uidNumber=0,cn=peercred,\
```