kleiner kommunikation
max@kleiner.com

1 OpenLDAP mit Delphi

Auch vor der Welt der Verzeichnisdienste macht Open Source nicht halt. Mit OpenLDAP besteht seit längerem ein mächtiger Dienst, welcher Benutzer wie Ressourcen systemübergreifend verwalten läßt. Lassen Sie sich im Folgenden vom Umfang des Servers wie der Schnittstellen überzeugen und die hohe Flexibilität zeigen. OpenLDAP ist dafür gebaut, stetige Änderungen einer Firmenstruktur laufend abdecken zu können. Dieser Bericht ist Anfang einer Serie, der mit OpenSSL und OpenGL eine Fortsetzung findet.

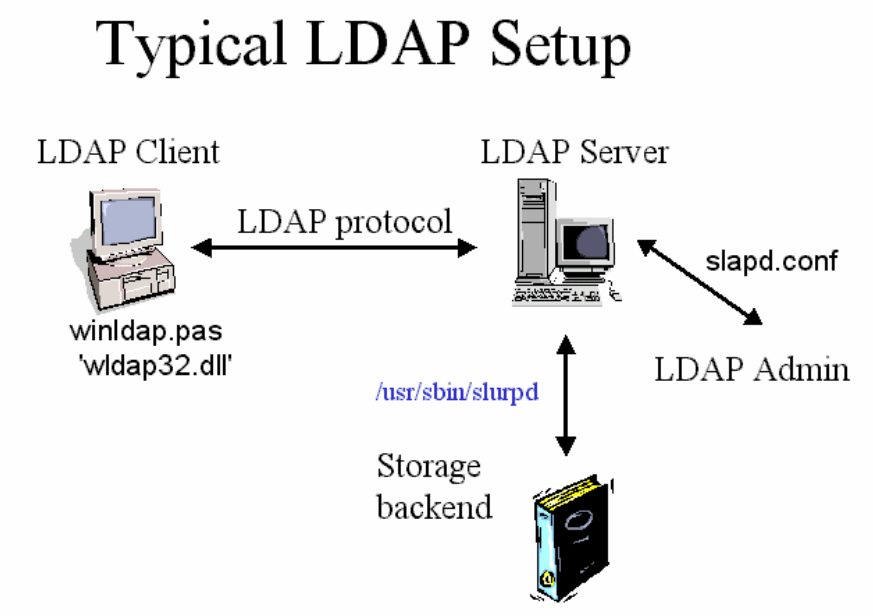
1.1 Integration der Import Unit

Jede Firma ist anders, jede Firma hat andere Anforderungen, jede Firma hat verschiedene Vorstellungen und Wünsche an einen Verzeichnisdienst. LDAP-basierte Verzeichnisdienste gewinnen nicht nur unter Linux und Unix, sondern auch in heterogenen Netzen zusehends an Bedeutung. LDAP ermöglicht einen flexiblen, sicheren, effizienten und skalierbaren Zugriff auf Benutzer- und Ressourceninformationen wie auch auf andere Daten wie bspw. binäre Bilder. An der EKON 12 hab ich gezeigt, wie man mit LDAP auf eine PKI und deren Public Keys Zugriff hat.

Mit OpenLDAP gibt es schon seit einiger Zeit eine robuste Implementierung des Lightweight Directory Access Protocol (LDAP). Das Open-Source-Projekt [i] ging vor einigen Jahren aus einem Server-Projekt der Universität Michigan hervor. OpenLDAP besteht aus einem skalierbaren Server mit passenden LDAP-Clients (siehe Abb.1) und unterstützt seit Version 2 endlich auch die Protokolle der LDAP-(Konzept-)Version 3.

OpenLDAP ist im Einsatz enorm flexibel und so verwundert es nicht, dass auch die praxisrelevanten Themen im Protokoll enthalten sind [ii] . Dazu gehören Replikation/Lastverteilung, Zugriffslisten (ACL), Verschlüsselung mit SSL/TLS und schließlich Authentisierung unter Verwendung von PAM.

Die TCP/IP-basierte Kommunikation mit Verzeichnissen erfolgt nach dem Client/Server-Modell, deshalb ist LDAP ein entsprechend standardisiertes Protokoll mit einer zugehörigen API.



//openldap_setup.tif

Abb. 1: Ein typischer LDAP Aufbau

Eine wesentliche Stärke vieler Verzeichnislösungen liegt in der Optimierung der Lese- und Suchoperationen. Das macht die Produkte insbesondere als Container für Daten interessant, die man deutlich häufiger abfragt als manipuliert. Weiterhin bieten LDAP Produkte gute Unterstützung bei der Implementierung verteilter Lösungen inklusive Replikation. Konsequenterweise ist der häufigste Einsatz von Verzeichnissen die unternehmensweite Verwaltung von Benutzerdaten und deren Berechtigungen, insbesondere bei Organisationen mit vielen Standorten. Verzeichnisse bilden einen elementaren Bestandteil von Single-Sign-On- und PKI-Lösungen (Public Key Infrastructure. Eine solche umfasst in der Regel die Ausgabe digitaler Zertifikate an Benutzer, welche ein Administrator im Verzeichnis verwaltet und geht nicht selten über die Verwaltung von Mitarbeiterdaten hinaus. Im Folgenden zeige ich exemplarisch diese Aufgabenstellung, die wir innerhalb von SecureCenter benötigen.

Gehen wir ans Eingemachte. Nachdem die der CD beigelegte Projektdatei *ldap2Client.dpr* kompiliert ist, werfen wir einen Blick auf die uses Anweisung der Formdatei. Mit Delphi lässt sich folgende DLL mit einer Interface Unit einbinden.

```
Description=Windows Lightweight Directory Access Protocol - WLDAP32.DLL
```

Die Import Unit *winldap.pas* stammt aus den Quellen von JEDI, die auch das vorliegende Beispiel nutzen. Eine weitere Unit existiert von Jens Geyer, der konzeptionell zeigt, wie man von Windows auf einen OpenLDAP (Linux) Server zugreifen kann. Die Delphi-Portierung von Teilen der WinLDAP.H (Header File) ist aktueller aber (noch) unvollständig.

Die folgende Anwendung in Abb. 2 selbst stammt ja in der Basis aus dem JEDI Projekt [iii], das auch eine entsprechende Import Unit bereithält und im Kern eine Online-Verbindung mit einem LDAP Dienst aufnimmt. Die Verbindungsparameter liest die Anwendung aus den Tiefen der Registry:

```
sREGKEY = '\Software\Microsoft\Internet Account Manager\Accounts\';
```

Hier sind die entsprechende URL sowie weitere Zugriffsattribute zu finden. Kleinere logische oder optische Anpassungen lassen sich zum Teil sogar während des laufenden Betriebs ändern. Der Code selbst ist in der Schlüsselstelle ziemlich flüssig aufgebaut und besteht aus den Segmenten Verbindung öffnen, authentifizieren und check, Suche initialisieren um dann durch die (gefundenen) hierarchischen Werte zu navigieren, die ich schlussendlich in einer `TListView` darstelle :

```
// open directory connection
pld:= ldap_open(PChar(sServer), iPort);
if Assigned(pld) then
try
  // authenticate anonymously
  LDAPCheck(ldap_simple_bind_s(pld, nil, nil));
  // perform search
  LDAPCheck(ldap_search_s(pld, PChar(sBase), LDAP_SCOPE_SUBTREE,
    PChar(sSearch), nil, 0, plmSearch));
  try
    // initialize results
    iRow:= 0;
    msResults.Clear;
    slAttribs.Clear;

    // loop thru entries
    plmEntry:= ldap_first_entry(pld, plmSearch);
    while Assigned(plmEntry) do begin
      // clear attributes
      slAttribs.Clear;
      // loop thru attributes
      pszAttr:= ldap_first_attribute(pld, plmEntry, pbe);
      while Assigned(pszAttr) do begin
        // store attribute
        iCol:= slAttribs.Add(pszAttr);
        // and get values
        ppcVals:= ldap_get_values(pld, plmEntry, pszAttr);
```

Die Directory Information Base (DIB) besteht aus einzelnen Einträgen den Entries, die zueinander in einer hierarchischen Beziehung stehen und ich im Code iterativ aufbaue. Eine Entry ist dabei eine Menge von Informationen über ein einzelnes Objekt. Entries (Verzeichniseinträge) repräsentieren in der Regel Objekte aus der Realität wie eben eine Zertifikatsliste oder ein Telefonbuch.

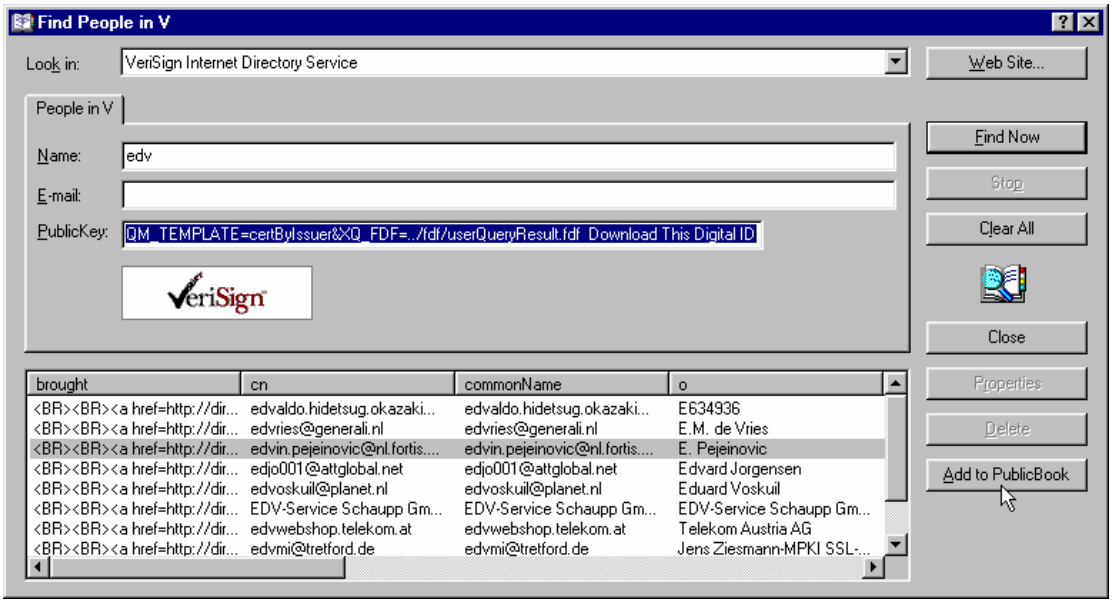
Entries wiederum setzten sich aus einer Sammlung von Attributen zusammen und jedes Attribut hat einen Attributtyp mit 1 oder n Werten, die letztlich mit folgendem Befehl zur Ausführung kommen:

```
ppcVals:= ldap_get_values(pld, plmEntry, pszAttr);
```

Größere Veränderungen im Code setzen meist Programmierkenntnisse im Schema voraus, sind aber möglich, auch wenn es sich um ganz neue Typen handelt. Dies ist realisierbar, denn ein LDAP-Schema definiert die Liste möglicher Typen von Einträgen (die man als Objektklassen bezeichnet) zusammen mit den verknüpften Attributen.

Diese Schema-Definitionen werden dann in Dateien gespeichert, die sich im Verzeichnis */schema* des LDAP Servers befinden, nebst seinem Hauptverzeichnis */data* als eigentlicher Datenspeicher[1].

Apropos Lizenz (Public): Der Lizenznehmer von OpenLDAP (aktuelle Version 2.4.12) ist berechtigt, unkompilierte oder integrierte Versionen uneingeschränkt zu verändern. Dabei sind die vorhandenen Urheberrechtshinweise auf allen Veränderungen und Kopien der Software beizubehalten.



/ openldap_client.tif

Abb. 2: Die Beispielanwendung mit VeriSign Zugriff auf Zertifikate

1.2 Die Logik des Servers

Wir kommen zur Installation des Servers. Die Installation ist für ein derartiges Gesamtpaket erstaunlich einfach. Dank der Setup Routine von ILEX [iv] hat man auch unter Windows eine Chance zu einem vollwertigen Server zu kommen. Es ist eine gepackte Version 2.2.29 die OpenLDAP, SASL, BDB, and OpenSSL beinhaltet. Ein Assistent begleitet den Nutzer beim Erstellen der Konfiguration, die in der *slapd.conf* eingetragen wird, und dem Einrichten des Maschinenzertifikats inklusive selbst signiertes CA als Root-Zertifikat (siehe Abb. 3):

Unter Linux sind die bekannten Schritte anzugehen:

```
# cd /usr/src/
# wget ftp://ftp.openldap.org/pub/OpenLDAP/openldap-release/openldap-
2.4.12.tgz
# tar zxvf openldap-2.4.12.tgz
# cd openldap-2.4.12
# ./configure
```

[1] Wobei LDAP kein Datenbank im relationalen Sinne ist!

```
# make
# make install
```

Im Weiteren registriert man zusätzlich den OpenSSL Support bezüglich Authentifizierung sofern erwünscht. Später folgende Änderungen lassen sich in der Konfigurationsdatei *slapd.conf* speichern. Darin paßt man Details wie den Datenbanktyp und das Schema, den Pfad und Namensraum von OpenLDAP (vom Server aus gesehen) und die Berechtigungen an.

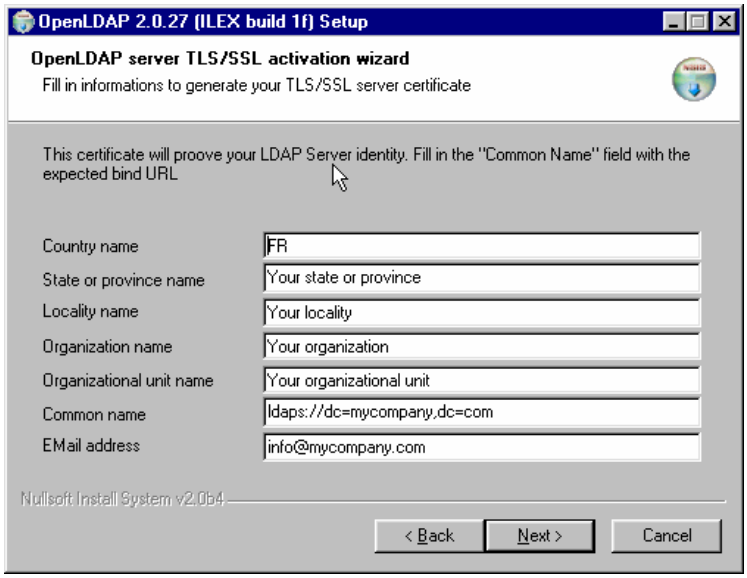
Um die Arbeit mit einem weiteren OpenLDAP Client zu vereinfachen, empfiehlt es sich die Datei */etc/openldap/ldap.conf* anzupassen. Hier können Basisangaben gespeichert werden, die später Tipparbeit ersparen.

Wie meistens lassen sich Open Source Produkte effizient (von Profis;)) aus der Kommandozeile oder Shells steuern. Das Verwalten des Servers kann aber auch innerhalb einer ordentlichen GUI vonstatten gehen, empfohlen wird das Tool LDAPAdmin, auf sourceforge beheimatet [v] .

Nach dem Start erscheint das zweiteilige Hauptmenü. Als erstes muss man in der Regel eine Verbindung aufbauen, die so aussehen kann:

`base: dc=softwareschule,dc=ch; host: www.softwareschule.ch; port: 389 etc.`

Die Navigation selbst ist intuitiv strukturiert und in 3 Masken hierarchisch aufgeteilt. Ein LDAP-Verzeichnis besitzt eine Baumstruktur. Die Informationen sind dann als Objekte und Eigenschaften von Objekten gespeichert, wobei oft verwandte Objekte zusammen in einem Zweig des Baums untergebracht sind. Das Tool zeigt die Baumstruktur im linken Teil des Hauptfensters an, die Attribute entweder (bei älteren Versionen) im rechten Teil oder nach Doppelklick auf das entsprechende Objekt in einem eigenen Dialogfenster.



//openldap_server_install.tif

Abb. 3: Der Assistent zur Konfiguration

Über die Masken des Tools lassen sich Informationen zu Verzeichnissen nur bedingt abfragen und darstellen. Um komplexe Abfragen und Gruppierungen zu ermöglichen, ist es möglich direkte Datenbankabfragen als Filter zu stellen, die das Tool entweder in einem Auflistfenster anzeigt oder in einer Datei speichert.

OpenLDAP unterstützt auch den Import von Daten über das LDAP Data Interchange Format (LDIF). Jeder neue Datensatz beginnt in diesem Format mit dem „distinguished name“ (dn), also dem Absoluten Pfad innerhalb eines LDAP Baumes. Im Anschluss daran lassen sich beliebige - im Schema vorhandene – Typen, Attribute und Werte importieren. Generell gilt bei LDIF Dateien die Schreibweise "attribut:wert" (Bsp.: dc=softwareschule,dc=ch).

Abschließend einige Bemerkungen zu LDAP und dem „Konkurrenten“ ADS. Obwohl die originäre Unterstützung von MS sich auf C++ und VB beschränkt, bieten sich hier auch dem ambitionierten Delphi-Entwickler Möglichkeiten für eigene Softwareentwicklungen. Die benötigten Informationen für

das ADS-Interface (ADSI) findet man im Win-SDK. Um das ADSI via COM mit Delphi zu verwenden benötigt man außerdem wie bei OpenLDAP eine Interface-Unit zum Windows-API.

Auch der Parallelbetrieb von OpenLDAP und Active Directory soll möglich sein. UCS AD Connector bspw. synchronisiert diese Objekte aus dem UCS-Verzeichnisdienst, der auf OpenLDAP basiert, mit Active Directory und umgekehrt. Die Technologie enthält auch eine bidirektionale Passwortsynchronisation. Ich wünsche einen herzhaften Winter und bis zur nächsten Folge „OpenSSL mit Delphi“.

Max Kleiner

Links:

[1] LDAP-Vortrag: http://www.softwareschule.ch/download/openldap_delphi.pdf

[2] LDAP-Sourcen: /download/openldap_delphi.zip

[3] LDAP Einführung: http://www.mitlinx.de/ldap/

---

[i] OpenLDAP-Projekt: http://www.openldap.org

[ii] RFC1777 - LDAProtocol: ftp://ftp.isi.edu/in-notes/rfc1777.txt

[iii] http://www.delphi-jedi.org/

[iv] www.ilex.fr/openldap

[v] http://ldapadmin.sourceforge.net/