Hintergründe und Basiswissen

Diese Seite schließt eine Lücke im Wiki. Sie soll Hintergrund und Basiswissen vermitteln, welches dabei hilft sich an der Entwicklung des invis-Servers zu beteiligen bzw. genutzte Techniken und Konfigurationen zu verstehen.

DNS-Server bind und LDAP

Schon von Anfang an setzen wir ein LDAP-Verzeichnis als Daten-Backend für den DNS-Server *bind* ein. Der zugrunde liegende Patch, welchen wir schon seit einiger Zeit selbst in die Software-Pakete einbauen müssen, wird inzwischen kaum noch weiter entwickelt. Statt dessen gehört der sogenannte DLZ (dynamic loadable zone) Treiber seit etwa openSUSE 11.2 (oder so) fest zum Funktionsumfang der openSUSE-Pakete. Die Nutzung dieses Treibers würde uns eine Menge Arbeit ersparen, da wir keine eigenen *bind* Pakete mehr im openSUSE-Buildservice pflegen müssten.

Schema-Definitionen

Jede Form der Datenspeicherung in einem LDAP-Verzeichnis bedarf der Definition von Attributen und Objektklassen (Tabellen- und Felddefinitionen einer SQL-Datenbank nicht ganz unähnlich). Diese Definitionen werden in Form sogenannter Schema-Dateien getroffen und haben Server-weit Gültigkeit. Zu finden sind Sie im Verzeichnis

```
/etc/openldap/schema
```

Der Aufbau der LDAP-Schemas (kein Witz, es schreibt sich so) „dlz.schema“ für den DLZ-Treiber und „dnszone.schema“ für den alten SDB-LDAP-Patch unterscheiden sich grundsätzlich voneinander.

Einige Unterschiede im Überblick:

- Allen Attributen und Objektklassen des DLZ-Schemas ist zur Vermeidung von Namensdopplungen die Silbe „dlz“ vorangestellt.
- Während das *DNSZone-Schema* lediglich die Objektklasse „dNSZone“ kennt über die alle DNS-Objekte dargestellt werden, verfügt das *DLZ-Schema* über Objektklassen für die verschiedenen DNS-Record-Typen. Dies dürfte etwas verwirrend wirken, da es für die verschiedenen DNS-Records auch entsprechende Attributdefinitionen gibt.
- Nach dem alten Schema beginnt jedes DNS-Objekt im LDAP mit dem Attribut „relativeDomainName“, welches als Schlüssel-Attribut angesehen werden kann. Im DLZ-Schema nimmt das Attribut „dlzRecordID“ diese Rolle ein. Eine Kennzeichnung des Objekts wird dabei erst über das Folge-Attribut erreicht: „dlzRecordID=05,dlzHostName=pc-buchhaltung,…“
- Während alle kennzeichnenden Informationen eines SOA-Records beim alten Schema im Attribut „sOARecord“ zusammengefasst wurden, kennt das DLZ-Schema für jeden Wert ein eigenes Attribut: z.B. „dlzSerial“, „dlzRefresh“, usw.
- Die Daten der MX-Records sind beim DLZ-Schema nicht Teil des SOA-Eintrages, sondern werden als eigene LDAP-Objekte angelegt.
- Das DLZ-Schema kennt weder Attribute noch Objektklassen für SRV-, mINFO, hINFO, SIG-, KEY-, sowie einige weitere DNS-Record-Typen. Diese können entweder über die Objektklasse

„dlzGenericRecord“ oder eigene Erweiterungen des Schemas realisiert werden. Für Letzteres ist mit „1.3.6.1.4.1.18420.1.3.X“ im Schema ein eigener Object-Identifier (OID) vorgesehen.
- Die meisten Attribute des DLZ-Schemas sind als Single-Values gekennzeichnet. D.H. gehören etwa zu einem Pointer-Record mehrere Hostnamen, können diese nicht in einem Objekt zusammengefasst werden, sondern es muss für jeden Hostnamen ein eigenes PTR-Objekt erzeugt werden.

Aufbau der LDAP-Objekte

Um die Unterschiede im Aufbau der jeweiligen LDAP-Struktur zu verdeutlichen werden im folgenden einige Standard-Einträge gegenübergestellt.

SOA-Record

- dNSZone-Schema

```
dn: relativeDomainName=@,ou=invis-net.loc,ou=forward,ou=zone.master,ou=DNS-
Server,dc=invis-net,dc=loc
dNSClass: IN
dNSTTL: 3600
mXRecord: 10 mail.invis-net.loc.
nSRecord: ns.invis-net.loc.
objectClass: dNSZone
objectClass: top
relativeDomainName: @
sOARecord: ns.invis-net.loc. root.invis-net.loc. 2006260342 3600 1800 604800
86400
zoneName: invis-net.loc
```

- DLZ-Schema

```
dn: dlzRecordID=11,dlzHostName=@,dlzZoneName=invis-
net.loc,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzSOARecord
dlzRecordID: 11
dlzHostName: @
dlzType: soa
dlzSerial: 2006260342
dlzRefresh: 3600
dlzRetry: 1800
dlzExpire: 604800
dlzMinimum: 86400
dlzAdminEmail: root.invis-net.loc.
dlzPrimaryns: ns.invis-net.loc.
dlzTTL: 10
```

```
dn: dlzRecordID=5,dlzHostName=@,dlzZoneName=invis-
server.loc,ou=zone.master,ou=DNS-Server,dc=invis-server,dc=loc
objectclass: dlzMXRecord
dlzRecordID: 5
dlzHostName: @
```

```
dlzType: mx
dlzData: mail
dlzPreference: 10
dlzTTL: 10
```

A-Record

- dNSZone-Schema

```
# A-Record des Servers
dn: relativeDomainName=invis5,ou=invis-
net.loc,ou=forward,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
aRecord: 192.168.220.10
dNSClass: IN
dNSTTL: 86400
objectClass: dNSZone
objectClass: top
relativeDomainName: invis5
zoneName: invis-net.loc
```

- DLZ-Schema

```
dn: dlzRecordID=14,dlzHostName=invis5,dlzZoneName=invis-
net.loc,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzARecord
dlzRecordID: 14
dlzHostName: invis5
dlzType: a
dlzIPAddr: 192.168.220.10
dlzTTL: 86400
```

PTR-Record

- dNSZone-Schema

```
dn: relativeDomainName=10,ou=220.168.192.in-
addr.arpa,ou=reverse,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
pTRRecord: invis5.invis-net.loc.
pTRRecord: ns.invis-net.loc.
pTRRecord: mail.invis-net.loc.
dNSClass: IN
dNSTTL: 86400
objectClass: dNSZone
objectClass: top
# Test ohne Leerzeichen
relativeDomainName:10
zoneName: 220.168.192.in-addr.arpa
```

- DLZ-Schema

```
dn: dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
```

```
objectclass: dlzHost
dlzHostName: 10
```

Die Begründung für diesen Eintrag finden Sie im nächsten Abschnitt.

```
dn: dlzRecordID=15,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 15
dlzHostName: 10
dlzType: ptr
dlzData: invis5.invis-net.loc.
dlzTTL: 86400
```

```
dn: dlzRecordID=16,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 16
dlzHostName: 10
dlzType: ptr
dlzData: mail.invis-net.loc.
dlzTTL: 86400
```

```
dn: dlzRecordID=17,dlzHostName=10,dlzZoneName=220.168.192.in-
addr.arpa,ou=zone.master,ou=DNS-Server,dc=invis-net,dc=loc
objectclass: dlzPTRRecord
dlzRecordID: 15
dlzHostName: 10
dlzType: ptr
dlzData: ns.invis-net.loc.
dlzTTL: 86400
```

LDAP-Struktur

Auch die Struktur des LDAP-Zweiges für den DNS-Server sollte sich mit Umstieg auf das DLZ-Schema ändern.

Bisher wurden beim invis-Server die DNS-Einträge nach Zone, Zonen-Typ und „Richtung“ (Vorwärts/Rückwärts) sortiert. Dies war in sich eher etwas unglücklich. Mit einem Umstieg werden wir uns an die Empfehlung des „Bind DLZ“ Projekts halten und lediglich nach Zone und Host sortieren. Eine Unterscheidung nach Vorwärts- oder Rückwärtsauflösung ist nicht generell nicht notwendig, da dies bereits über die Zone selbst klar wird.

Der Basisknoten bleibt hingegen unverändert:

```
ou=DNS-Server,ou=domain,ou=tld
```

Dem wurde beim alten Schema noch der Zonentyp (Master oder Slave), die Richtung der Auflösung sowie der Zonenname vorangestellt.

```
ou=domain.tld,ou=forward,ou=zone.master,...
```

Zukünftig wird sich dies Ändern. Zunächst ist der Zonenname hier mit einem eigenen Attribut gekennzeichnet, dafür eine eigene „Organizational Unit“ (ou) in den DN (Distinguished Name) eines DNS-Objektes einzufügen ist also obsolet. Weiterhin entfällt wie bereits erwähnt die Unterscheidung zwischen Vorwärts- und Rückwärtsauflösung, etwas worauf wir auch schon früher hätten verzichten können. Zu guter Letzt fügen wir der DLZ-Empfehlung folgend allerdings bei Bedarf den Hostnamen als zusätzlichen Knoten ein. Dies ist dann sinvoll, wenn etwa zu einem Host mehrere Einträge, seien es A- oder PTR-Records, folgen:

Also, entweder:

```
dlzZoneName=domain.tld,ou=zone.master,....
```

oder:

```
dlzHostname=host,dlzZoneName=domain.tld,ou=zone.master,....
```

Letzteres eben im Falle mehrerer Records zu einem Host.