invis-Classic zu invis-AD

An dieser Stelle sollte von vorne herein klar sein, dass es kein direktes Upgrade von einem invis-Classic auf einen invis-AD gibt und geben kann. Es gibt aber eine Reihe von Möglichkeiten Daten aus einer alten in eine neue Installation zu migrieren. Überlegen sie sich aber gut, ob Sie sich das antun wollen. Sie müssen mit vielen Stunden mühsamer Bastelarbeit rechnen, die zwar lehrreich ist aber Möglichkeiten für viele Fehler bietet. Ich habe die Erfahrung gemacht, dass es bei Installation bis 25 Benutzerkonten schneller geht alles von Grund auf neu zu machen.

Im Idealfall wird ein neuer invisAD auf neuer Hardware aufgebaut. Möglich aber um Längen aufwändiger ist die Neuinstallation auf gleicher Hardware. In letzterem Fall müssen zuvor alle relevanten Daten der alten Installation gesichert werden.

Zu den zu sichernden Daten zählen:

- Das vollständige LDAP Verzeichnis in Form eines LDIF-Dumps
- Das gesamte „/etc“ Verzeichnis
- Alle Datenbanken inklusive der Dokuwiki Daten
- Alle Nutzdaten des Fileservers
- Den gesamten Mailbestand.

Alleine für die Sicherung müssen Sie je nach Datenmenge mit vielen Stunden Arbeit und vor allem Wartezeit rechnen. Da ist es eher ratsam sich für die Neuinstallation wenigstens neue Festplatten zu gönnen und die Platten des alten Servers für die Dauer die Migration in einen anderen PC einzubauen.

Wie auch immer, aus eigener Erfahrung empfehle ich **immer** komplett neue Hardware für die Neuinstallation.

Es gilt vorab eine weitere grundsätzliche Entscheidung zu fällen. Der Aufwand LDAP-Informationen aus einem OpenLDAP Dump zu extrahieren und für den Import in ein AD umzubauen ist mühsam und fehleranfällig. Wenn es sich beim zu migrierenden invis-Server um eine Installation mit nur wenigen Benutzern und überschaubarem Umfang an IP-Geräten im Netz handelt, ist es einfacher und schneller die Daten in der Neuinstallation über das invis-Portal neu einzugeben. Die Grenze würde ich bei ca. 20 Benutzern ziehen.

Nachfolgend Anleitungen und Tipps zur Bewältigung der Migration. Eine genaue Anleitung wie die oben beschriebene invisAD → invisAD Migration wird es hier nicht geben.

LDAP Informationen migrieren

Zu den LDAP-Informationen zählen Benutzerkonten, deren zugehörige Email-Konten Informationen und die Daten von DNS- und DHCP-Server.

Benutzerkonten

Beginnen wir mit einer ernüchternden Information. **Lassen Sie es sein!**

Samba bietet mit der Funktion **classic-upgrade** eine Möglichkeit eine Windows-Domäne auf Basis von Samba3 (NTLM Domäne) unabhängig vom verwendeten Backend (ldap, tdbsam usw.) in eine AD-Domäne zu migrieren. Diesen Weg habe ich bereits einmal erfolgreich beschritten. Ziel der damaligen Migration war allerdings kein invis-Server sondern ein alleinstehender Active-Directory Domain-Controller.

invis-Server nutzen OpenLDAP (Classic) bzw. den LDAP-Dienst eines Active-Directories für weit mehr als nur die Speicherung von Benutzerinformationen. Einerseits erweitern wir das AD mit zahlreichen Schema-Erweiterungen, diese müssten nach einem Classic-Upgrade alle händisch eingepflegt werden. Andererseits muss man sich der Tatsache bewusst sein, dass die Benutzerverwaltung mit AD gänzlich anders aufgebaut ist als auf dem klassichen Server. Ein invis-Classic legt grundsätzlich POSIX-kompatible Benutzerkonten an, die um Windows-Attribute erweitert wurden. Ein invisAD hingegen legt Windows-Benutzerkonten an, die um POSIX-Attribute erweitert werden. Auf einem invisAD Server nutzen wir gänzlich andere UID- und GID-Bereiche als auf dem invis-Classic. Daraus können sich nach einer solchen Migration Probleme im laufenden Betrieb ergeben.

Wer dennoch diesen Weg beschreiten möchte findet im Samba Wiki eine Anleitung für den Upgrade-Prozess.

Vorbereitend müssen Sie dennoch eine volle invisAD-Installation durchführen. Stoppen Sie vor dem Classic-Upgrade den Samba-Dienst, sichern Sie die Samba-Konfiguration unter

```
/etc/samba
```

und erstellen Sie ein Backup des neuen leeren ADs:

```
linux:~ # adbackup
```

Ist die Migration per Classic-Upgrade gelungen, müssen Sie alle Daten- und Schema-Erweiterungen händisch ins neue AD integrieren. Auch das habe ich bereits einmal durchgeführt, die entsprechende Anleitung folgt.

Früher oder später wird der invisAD-Server auch ein Script zum massenhaften anlegen neuer Benutzerkonten mitbringen, was dann auch eine gewisse Erleichterung darstellt.

Der Vorteil dieser mühsamen Methode sollte allerdings nicht unerwähnt bleiben. Die Benutzerprofile der Domänenbenutzer funktionieren auch nach der Migration noch, da sich die Dmomain-SID hierbei nicht ändert. Eine vollständige Neuinstallation bedeutet neue Benutzer-Konten auch wenn sich Benutzernamen und Passwörter nicht ändern. Das bedeutet, dass alle Client-PCs der Domäne neu beitreten und alle Benutzerprofile neu aufgebaut werden müssen. Ein wenig googlen fördert zwar auch Tools zur Profilmigration zu Tage, allerdings kann ein wenig „Tabula Rasa“ manchmal nicht schaden.

DHCP und Mailkonten Daten migrieren

LDIF-Dateien zerlegen und sortieren

Bei der Migration von LDAP-Datenbeständen müssen immer wieder LDAP-Dateien von unnötigem

Ballast befreit, zerlegt und neu zusammen gesetzt werden.

Export

Zunächst muss der gesamte Inhalt eines eines openLDAP-Verzeichnisses im LDIF-Format exportiert werden:

```
linux:~ # slapcat > allldap-01022014.ldif
```

Bereinigen

Aufgrund der definierten Struktur von LDIF Dateien ist dies nicht einfach, beispielsweise werden lange Zeilen nach 79 Zeichen umbrochen und die nachfolgenden Zeilen dadurch kenntlich gemacht, dass sie mit einem Leerzeichen beginnen. D.h. beim löschen überflüssiger Zeilen, können Fragmente der Zeilen übrigbleiben, da Tools wie ***grep*** zusammenhängende Zeilen nicht erkennen können.

Speziell bei der Migration von openLDAP zu Active-Directory müssen alle Zeilen mit folgenden Einträgen gelöscht werden:

- entryUUID
- creatorsName
- createTimestamp
- entryCSN
- modifiersName
- modifyTimestamp.

Alle genannten Einträge können in einem Schritt entfernt werden:

```
linux:~ # cat allldap-01022014.ldif |grep -v ^entryUUID:|grep -v
^creatorsName:|grep -v ^createTimestamp:|grep -v ^entryCSN:|grep -v
^modifiersName:|grep -v ^modifyTimestamp > cleaned.ldif
```

Schaut man sich die Ergebnisdatei genauer an, kann es sein, dass darin Zeilen vorkommen, die mit einem Leerzeichen beginnen, in denen evtl. nur wenige Zeichen vorkommen und/oder die in keinem Zusammenhang mit der vorhergehenden Zeile stehen. Diese Zeilen sind Fragmente des erwähnten Zeilenumbruchs nach 79 Zeichen und müssen im zweiten Schritt entfernt werden.

In einem Anwendungsfall waren das Zeilen die lediglich „ c“, „ oc“ und „ loc“ enthielten.

Entfernen der gefundenen Zeilenfragmente:

```
linux:~ # cat cleaned.ldif | grep -v "^ oc$" | grep -v "^ loc$"  | grep -v "
c$" > cleaned2.ldif
```

Zerlegen

Alle LDAP-Objekte sind innerhalb einer LDIF-Datei durch Leerzeilen voneinander getrennt. Jetzt muss

die bereinigte Datei anhand der Leerzeilen in einzelne Dateien zerlegt werden.

Verwendet wird das Tool ***csplit***. Allerdings scheint es ein Problem zu sein einen wirklich passenden regulären Ausdruck für die Leerzeilen der LDIF-Dateien zu finden. Der unten gezeigte Ausdruck hat funktioniert. Er passt schlicht dann, wenn die Zeile **nichts** enthält. Komischerweise scheint sie nicht einmal einen „Zeilenumbruch“ zu enthalten. Seltsam!

```
linux:~ # csplit -ks ../cleaned2.ldif '/^$/' '{*}'
```

Danach liegen im aktuellen Arbeitsverzeichnis alle Objekte als einzelne Dateien vor. Benannt sind sie nach dem Schema „xxN“, dabei ist N eine laufende Nummer. Da es je nach Größe der alten Installation unter Umständen mehrere Hundert oder Tausend Einzeldateien sein können, sollten Sie sich dafür ein gesondertes Verzeichnis anlegen.

Anwendungsspezifisch zusammensetzen

Im nächsten Schritt müssen aus den Einzeldateien wieder zusammengehörige LDIF-Dateien erzeugt werden. Am Beispiel der im LDAP gespeicherten Daten für CorNAz.

```
linux:/pfad/zu/einzeldateien # cat  `grep -l "dn: fsp" ./*` >>
../cornaz.ldif
```

...und des DHCP-Servers:

```
linux:/pfad/zu/einzeldateien # cat  `grep -l "=DHCP-" ./*` >> ../dhcp.ldif
```

Von OpenLDAP zu Active Directory

Um alte invis-Server Installationen von OpenLDAP nach Active-Dircetory zu migrieren sind vor allem die Daten von CorNAz (fetchmail) und die des DHCP-Servers wichtig und für die Migration geeignet. DNS-Daten und Benutzerkonten müssen manuell im AD angelegt, bzw. letztere mit Hilfe von ***samba-tool*** migriert werden.

Zu beachten ist bei der Migration, dass die LDAP-Objekte des DHCP-Servers und von CorNAz an anderer Stelle im LDAP-DIT liegen und im AD statt des Attributes „ou“ (organizational unit) in der Regel „CN“ (container) verwendet wird. Beim isc-DHCP Server worden überdies die die Namen von Objektklassen und Attributen um die Vorsilbe „isc“ ergänzt, damit es nicht zu Konkurrenzen mit Objektklassen und Attributbezeichnungen des Microsoftschen DHCP-Serves kommt.

Weitere Probleme können dadurch verursacht werden, dass Microsofts AD hinsichtlich der Bezeichnungen von Objektklassen und Attributen an manchen Stellen Case-Sensitiv ist und an anderen nicht.

Schlussendlich bedeutet dies, dass die bereits generierten LDIF-Dateien mit „Suchen & Ersetzen“, sowie reichlich Handarbeit für den Active-Directory Import umgebaut werden müssen. Aufgrund des zuvor beschriebenen Zeilenumbruchs nach 79 Zeichen im LDIF-Format kann „Suchen & Ersetzen“ nicht zu 100% funktionieren. Auch müssen eventuell nach dem Umbau der Dateien neue

Zeilenumbrüche eingefügt werden.

Speziell für Informationen des invis-Servers haben wir im Activ-Directory einen zusätzlichen Container-Knoten „cn=invis-server“ angelegt. Er beherbergt die Daten des DHCP-Servers, von CorNAz (fetchmail/postfix), des invis-Portals und weitere.

```
DN: cn=invis-server,dc=domain,dc=tld
```

Weiterhin kennt Active-Directory nicht das Attribut „structuralObjectClass“. Dies muss aus allen erzeugten LDIF-Dateien entfernt werden:

```
linux:~ cat cornaz.ldif |grep -v structuralObjectClass > cornaz2.ldif
```

CorNAz

Die Daten der Email-Konten und Adressen der Server-Benutzer liegen jetzt nicht mehr jeweils als Objekte unterhalb der Benutzerkonten sondern im oben erwähnten Container:

```
DN: cn=username,cn=AdditionalUserInformation,cn=invis-
server,dc=domain,dc=tld
```

Entsprechend müssen die DNs der einzelnen Einträge umgebaut werden. Zuvor muss allerdings für jeden Benutzer ein neuer Container generiert werden. Dies kann mit Hilfe eines einfachen Shell-Scripts aus dem ursprünglichen LDAP-Dump geschehen.

Zunächst müssen per Suchen & Ersetzen, sowie händischer Nacharbeit folgende Attribute bzw. Zeichenketten geändert werden:

- fspExtMailAddress= → CN=
- fspLocalMailAddress= → CN=
- uid= → CN=

Das einbezogene „Gleichheitszeichen sorgt dafür, dass die Attribute lediglich im DN der Objekte geändert werden. Sie bleiben als Attribute im Objekt erhalten.

Da sich der Pfad zum Objekt im DIT ändert, muss der DN noch im Ganzen geändert werden:

Aus:

```
DN: ...,ou=users,ou=Benutzerverwaltung,....
```

wird

```
DN: ...,CN=AdditionalUserInformation,CN=invis-server,....
```

Speziell hier wird es mit automatischem Suchen & Ersetzen schwer, da der DN in der LDIF Datei meist länger als 79 Zeichen ist und der Zeilenumbruch in Abhängigkeit der Länge des Benutzernamens an unterschiedlichen Stellen erfolgt. Die gesamte Datei ist komplett händisch nachzuarbeiten.

Da sich auch das Objekt-benennende Attribut (RDN) geändert hat, ist dieses Attribut jeweils als

eigene Zeile in jedes Objekt einzufügen.

So wird beispielsweise aus „fspExtMailAddress=user@domain.de“ → „CN=user@domain.de“, entsprechend ist im Objekt die Zeile:

```
CN: user@domain.de
```

einzufügen.

Achtung: *Es ist darauf zu achten, dass es zwei unterschiedliche Objekttypen gibt: „fspExtMailAddress“ und „fspLocalMailAddress“. Entsprechend unterscheidet sich die einzufügende Zeile. Sie enthält mal die externe und mal die interne Email-Adresse des Users.*

Mit folgendem Shell-Script eine weitere LDIF-Datei zur Erstellung der neuen Benutzer-Knoten generiert werden:

createadnode

```
#!/bin/bash

cornazbase="CN=AdditionalUserInformation,CN=invis-
server,DC=localdomain,DC=tld"
filename=$1

users=(`cat $filename |grep "fspLocalMailAddress:" |cut -d " " -f 2 |cut -d
"@" -f1|sort -u`)

for user in ${users[*]}; do
    # DN erstellen
    dn="dn: CN=$user,$cornazbase"
    # Anzahl der Zeichen im DN zählen
    dncount=`echo $dn | wc -c`
    # DN zerlegen, wenn laenger als 79 Zeichen
    if (( $dncount > 79 )); then
        dn1=`echo $dn|cut -c 1-79`
        dn2=`echo $dn|cut -c 80-`
        echo "$dn1"
        if [[ -n $dn2 ]]; then
            echo " $dn2"
        fi
    else
        echo "$dn"
    fi
    #LDIF ergaenzen
        echo "cn: $user"
        echo "description: Email-Konten von $user"
        echo "name: $user"
        echo "objectclass: top"
        echo "objectclass: container"
        echo
```

```
done
```

Im Script ist die Variable „$cornazbase“ an die eigene Umgebung anzupassen. Der Aufruf des Scripts sieht wie folgt aus:

```
linux:~ # createadnode /pfad/zu/allldap-01022014.ldif >
/pfad/zu/cornaznodes.ldif
```

Legen Sie vor dem Import der generierten LDIF-Dateien unbedingt ein Backup Ihres Active-Directories an. Ein sicherungsscript ist im invis-Server AD bereits enthalten:

```
linux:~ # adbackup
```

Jetzt muss zunächst die soeben erzeugte LDIF-Datei mit den Benutzerknoten importiert werden:

```
linux:~ # ldbadd -v -H /var/lib/samba/private/sam.ldb
/pfad/zu/cornaznodes.ldif
```

Danach können die CorNAz-Objekte importiert werden:

```
linux:~ # ldbadd -v -H /var/lib/samba/private/sam.ldb /pfad/zu/cornaz2.ldif
```

Aufgrund der vielen Handarbeit bei der Erstellung der Import-Datei sind hier Importfehler keine Seltenheit. Achten Sie darauf, dass der Import sofort beginnt und beim ersten fehlerhaften Objekt endet. Tritt ein Fehler auf, erzeugen Sie eine Kopie der Datei unter neuem Namen und löschen alle Zeilen bis zum ersten fehlerhaften Objekt aus der Datei, beheben Sie den Fehler und starten Sie den nächsten Import-Versuch. usw.

Hinweis: *Es ist definitiv kein Fehler nach Abschluss jeder Import-Etappe eine Sicherung des gesamten ADs anzulegen.*

isc-DHCP Server

Etwas weniger komplex ist der Umbau der LDIF-Datei für den Import der DHCP-Server Daten. Hier müssen ebenfalls die DNs umgebaut und weiterhin die Attribute und Objektklassen des isc-DHCP Servers umbenannt werden.

Letzteres kann mit Hilfe von ***sed*** erfolgen:

```
linux:~ # cat dhcp.ldif | sed /dhcp[A-Z]/s/dhcp/iscDhcp/g > dhcp2.ldif
```

Der DN der Objekte wird wie folgt umgebaut.

Aus:

```
DN: cn=hostname,cn=DHCP Config,ou=DHCP-Server,dc=localdomain,dc=loc
```

wird:

```
DN: CN=hostname,CN=DHCP Config,CN=DHCP-Server,CN=invis-
server,CN=localdomain,CN=loc
```

Abgesehen vom Problem mit den Zeilenumbrüchen lässt sich hier viel mehr mir „Suchen & Eretzen“ erreichen. In der Praxis ist es so, dass in der ursprünglichen Datei die DN-Zeile eigentlich nie umbrochen ist. Durch Einfügen des Teils “,CN=invis-server„ werden aber Umbrüche nach 79 Zeichen notwendig. Dazu ist die Datei händisch durchzugehen.

Der Import erfolgt auf die gleiche Weise wie bei den CorNAz-Daten:

```
linux:~ # ldbadd -v -H /var/lib/samba/private/sam.ldb /pfad/zu/dhcp2.ldif
```

Konfigurationen migrieren

Datenbanken migrieren

Email Bestand migrieren

Klassische invis-Server verwenden entweder Dovecot- oder Cyrus-IMAP Server als Postfachspeicher. invis-AD dagegen Dovecot-IMAP oder Zarafa. Da Zarafa selbst mit dem Zarafa-Gateway einen IMAP-Dienst mitbringt ist eine IMAP-zu-IMAP Synchronisation der ideale Weg.

Im Netz lässt sich auch ein einfaches Tool zur direkten Synchronisation eines Maildirs hin zu Zarafa finden, leider hat dies ein zwei Nachteile:

1. Es ist sehr langsam (Test mit einem 18GB Postfach war nach 16 Stunden noch nicht fertig)
2. Es verwendet für den Import Zarafas „dagent“. Damit erhält jede importierte Email einen neuen aktuellen Zeitstempel. Dieses Verhalten ist ein absolutes „no go“.

Für die direkte „IMAP zu IMAP“ oder „Maildir zu IMAP“ Synchronisation eignet sich das Tool ***offlineimap*** am besten. Es ist überdies in gängigen Linux-Distributionen enthalten.

Speziell bei der Migration hin zu Zarafa muss zwischen unterschiedlich benannten Mail-Ordnern synchronisiert werden, dies wird zusätzlich klompliziert, da Zarafa kein UTF-8 verwendet.

Für ***offlineimap*** müssen also Übersetzungsregeln erzeugt werden. Auf dem Quell-Server muss im Home-Verzeichnis des Benutzers „root“ eine Datei namens

```
.offlineimaprc
```

angelegt werden. Nachfolgend ein paar Beispiele:

Beispiel: Dovecot zu Zarafa

```
[general]
accounts = privat
```

```
maxsyncaccounts = 1
[Account privat]
remoterepository = Alt
localrepository = Neu
[Repository Alt]
type = IMAP
remotehost = localhost
remoteuser = heinzb
remotepass = password
ssl = no
maxconnections = 1
readonly = true
nametrans = lambda foldername: re.sub('^INBOX.Sent$' ,'Gesendete Objekte',
        re.sub('^INBOX.Trash$' ,'Gel&APY-schte Objekte',
        re.sub('^INBOX.Drafts$', 'Entw&APw-rfe',
        re.sub('^INBOX.Junk$', 'Junk E-Mail',
        foldername))))

[Repository Neu]
type = IMAP
remotehost = 192.168.42.10
remoteuser = heinzb
remotepass = p@ssw0rd
ssl = no
maxconnections = 1
```

Die im Beispiel gezeigten „nametrans“ Regeln sind unter Anderem auch von der Konfiguration des Namespaces „private“ abhängig. Es sollte bekannt sein, ob hier der “/„ (Slash) oder der “.„ (Punkt) als Verzeichnistrenner verwendet wird und ob die IMAP-Spezial-Ordner (Trash, Sent, Drafts usw.) auf gleicher Verzeichnisebene mit der INBOX oder eine Verzeichnisebene tiefer liegen. Das Beispiel geht von einem „Punkt“ als Trenner und untergeordneten Spezial-Ordnern aus.

Letztlich werden ein paar Synchronisationsversuche notwendig sein, bis es reibungslos läuft.

Auf Quellseite muss die Direktive:

```
readonly = true
```

gesetzt werden, da ***offlineimap*** ansonsten eine bidirektionale Synchronisation durchführt.

Es kann nicht schaden, vor der Synchronisation leere IMAP-Ordner zu entfernen. Im Falle von „Maildirs“ auf der Quell-Seite kann das einfach im Dateimanager erfolgen. Im Falle von Cyrus ist ein angeschlossener Mail-Client zum Aufräumen erforderlich.

Fileserver-Datenbestand migrieren