# Migrationsstrategien
# Shib IdP 2.4.x → IdP 3.2.x

Wolfgang Pempe, DFN-Verein
pempe@dfn.de

DFN-AAI IdP-Workshop,
7. Juli 2016, TU Kaiserslautern

# Vorbemerkungen (1)

- Vom Upgrade einer existierenden IdP 2.x Installation ist dringend abzuraten!
- Shib IdP 3.x benötigt eine aktuelle Systemungebung, siehe hierzu https://wiki.aai.dfn.de/de:shibidp3#systemumgebung
- Bestehende IdP 2.x Konfiguration kann nur in Teilen übernommen werden
- Selbst mit den noch halbwegs kompatiblen Attribute Filter und Attribute Resolver Konfigurationen wird das System mit veralteter Sytnax und Parametern belastet.

# Vorbemerkungen (2)

- Ein allgemeingültiges *Cookbook* für die Migration eines produktiven IdP existiert nicht
- Die Durchführung hängt stark von den lokalen Gegebenheiten ab
  - Technische Infrastruktur
  - Cluster
  - Plugins
  - Eigenentwicklungen
  - u.a.m.
- Insofern können die folgenden Folien allenfalls als unverbindliche Anregungen verstanden werden, um eine passende Strategie für eine Migration zu entwickeln.

# IdPv3 Instanz aufsetzen und testen

- Vorarbeiten, Installation und Konfiguration gemäß Doku im DFN-AAI und Shibboleth Wiki
  https://wiki.aai.dfn.de/de:shibidp3
- IdP in Testföderation anmelden und ausführlich testen
- Gegenüber Version 2 sind u.a. folgende Punkte zu beachten (s.a. Release Notes):
  - Bestehende Datenbank(tabellen) für User Consent (uApprove) können **nicht** übernommen werden:
    https://wiki.shibboleth.net/confluence/display/IDP30/ConsentConfiguration
  - ComputedId ist deprecated. Falls etwa für ePTID benötigt, siehe
    https://wiki.aai.dfn.de/de:shibidp3extdataconnector
  - Persistent Id wird nicht mehr wie ein Attribut behandelt, vgl.
    https://wiki.aai.dfn.de/de:shibidp3storage#konfiguration_der_persistent-id
  - Änderung Tabellenschema für Stored Id (→ Folien am Schluss)

# Variante 1: Selber Server

- Deployment auf selbem Server wie bisheriger Produktiv-IdP
- Voraussetzung: Systemumgebung für IdPv3 geeignet

1. IdP-Verzeichnis des neuen IdP z.B. via tar in ein Archiv packen, dieses auf den Produktivserver übertragen und in ein paralleles Verzeichnis zum bisherigen IdP entpacken, z.B. /opt/shibboleth-idp-new o.ä.
2. In diesem neuen Verzeichnis alle Pfadangaben (z.B. mittels `rpl`) und die Entity ID auf die des Produktiv-IdP ändern (→ ggf. Angaben in MD-Verwaltung rechtzeitig anpassen/ergänzen)
3. Verzeichnisse umbennen, so dass der im Tomcat konfigurierte Pfad für den IdP nun auf die neue Instanz weist, also z.B.
   /opt/shibboleth-idp → /opt/shibboleth-idp-old
   /opt/shibboleth-idp-new → /opt/shibboleth-idp
4. Anschließend mittels bin/build.sh das WAR File neu bauen
5. Tomcat neu starten und den Neustart via tail -f im Logfile verfolgen

# Variante 1- Diskussion

- Nachteile:
  - Downtime (einige Minuten)
- Vorteile:
  - Falls Probleme auftreten, kann die Aktion durch Rück-Umbenennen des IdP-Verzeichnisses schnell wieder rückgängig gemacht werden
- Weiterhin zu beachten:
  - nicht vergessen, das für die Darstellung von JSP-Seiten erforderliche (z.B. Status-Abfrage) JSTL Jar-File im /lib Verzeichnis des lokalen Tomcat abzulegen, siehe hierzu https://wiki.aai.dfn.de/de:shibidp3prepare#tomcat_8 (weiter unten)
  - Datenbank für Persistent / Stored Id zuvor anpassen, siehe Folien am Schluss

# Variante 2: Neuer Server, gleicher FQDN

- Deployment auf neuem Server, Umstellung über Änderung des DNS-Eintrags (rechtzeitig TTL herabsetzen!)

1. Auch hier muss die IdP-Konfiguration entsprechend angepasst werden (Entity ID, Pfade). Sollte(n) sich auch das/die Zertifikat(e) ändern und/oder neue Binding URLs dazukommen, muss das natürlich rechtzeitig(!) in den Föderationsmetadaten eingetragen werden (ggf. DFN-AAI Hotline kontaktieren).
2. Bei der Übernahme des Hostnames des bisherigen Produktiv-IdP müssen die entsprechenden Änderungen natürlich systemweit erfolgen, nicht nur in der IdP-Konfiguration
3. Mittels ./bin/build.sh das WAR File neu generieren und Tomcat neu starten
4. Dann muss der neue Host entweder die IP Adresse(n) des bisherigen IdP Hosts übernehmen, oder aber der zugehörige DNS-Eintrag muss geändert werden.

# Variante 2 - Diskussion

- Nachteile:
  - Höherer Aufwand
- Vorteile:
  - Relativ nahtloser Übergang, falls alles gut geht
- Weiterhin zu beachten:
  - Auch hier darauf achten, dass die bestehenden Datenbankeinträge für die persistent IDs weitergenutzt werden. Entweder der Inhalt der DB wird gedumpt, angepasst und in eine neue DB eingespielt, oder die bestehende DB wird – entsprechend modifiziert – weitergenutzt. Siehe hierzu die folgenden Folien

# Variante 3: Neuer Server, neuer FQDN

- Deployment auf neuem Server, vom alten IdP wird nur die Entity ID übernommen

1. In IdP-Konfiguration die Entity ID auf die des bisherigen IdP ändern (./conf/idp.properties), WAR File neu generieren (./bin/build.sh) und Tomcat neu starten
2. In der Metadatenverwaltung beim neuen IdP die Binding URLs und Zertifikate des alten IdP mit eintragen
3. Den Eintrag für den alten IdP komplett löschen und den neuen IdP unter der alten Entity ID abspeichern

   (bei den Schritten 2 und 3 am besten die DFN-AAI Hotline kontatieren)
4. Am nächsten Tag können die alten Binding URLs und Zertifikate entfernt werden
5. Den alten IdP abschalten / vom Netz nehmen

# Variante 3 - Diskussion

- Nachteile:
  - Metadaten-Frickelei
- Vorteile:
  - Relativ nahtloser Übergang, falls alles gut geht
- Weiterhin zu beachten:
  - Auch hier darauf achten, dass die bestehenden Datenbankeinträge für die persistent IDs weitergenutzt werden. Entweder der Inhalt der DB wird gedumpt, angepasst und in eine neue DB eingespielt, oder die bestehende DB wird – entsprechend modifiziert – weitergenutzt. Siehe hierzu die folgenden Folien

# Datenbanktabelle für Persistent ID

Die Tabellendefinition sieht in etwa so aus:

```
CREATE TABLE shibpid (
    localEntity VARCHAR(255) NOT NULL,
    peerEntity VARCHAR(255) NOT NULL,
    persistentId VARCHAR(50) NOT NULL,
    principalName VARCHAR(50) NOT NULL,
    localId VARCHAR(50) NOT NULL,
    peerProvidedId VARCHAR(50) NULL,
    creationDate TIMESTAMP NOT NULL,
    deactivationDate TIMESTAMP NULL,
    PRIMARY KEY (localEntity, peerEntity, persistentId)
);
```

Die Zeile "`PRIMARY KEY`" ist neu ...

# Datenbanktabelle ändern, Variante 1

- **In jedem Fall zuvor DB-Inhalt sichern!**
  (z.B. via mysqldump)
- Primary Key kann zur Laufzeit eingefügt werden:

```
mysql> USE shibpid_db;

mysql> ALTER TABLE shibpid ADD PRIMARY KEY(localEntity,
peerEntity, persistentId);
```

- NB: DB-Namen (hier: shibpid_db) ggf. anpassen
- Zu `ALTER TABLE` siehe
  http://dev.mysql.com/doc/refman/5.5/en/alter-table.html
- Zum Thema mysqldump siehe
  https://dev.mysql.com/doc/refman/5.5/en/mysqldump.html

# Datenbanktabelle ändern, Variante 2

1. Dump der bestehenden Datenbank, z.B.
   `mysqldump DB_NAME > dump.sql`
   (`DB_NAME` durch tatsächlichen Name der Datenbank ersetzen)
2. Von der Datei mit dem Dump vorsichtshalber eine Sicherungskopie anlegen!
3. Dann im Dump die Zeile mit dem `PRIMARY KEY` Eintrag ergänzen (bitte auf korrekte Syntax achten!), siehe Beispiel oben sowie im Shibboleth Wiki und dem DFN-AAI Wiki
4. Dump zurückspielen
   `mysql DB_NAME < dump.sql`

Zum Thema mysqldump siehe auch
https://dev.mysql.com/doc/refman/5.5/en/mysqldump.html

# Vielen Dank für Ihre Aufmerksamkeit!

# Fragen? Anmerkungen?

## Kontakt


www: https://www.aai.dfn.de

eMail: hotline@aai.dfn.de

Tel.: +49 711 63314 215