zpub – Zentrales Publikationssystem

Technische Dokumentation

Joachim Breitner
Thomas Breitner

zpub-Version 0.6.2-15-g67911bd

2016

Inhaltsverzeichnis

Versionshistorie

Anmerkung

Siehe auch https://github.com/nomeata/zpub/releases

Version 0.6.2

Veröffentlicht am 7.2.2016

- Packaging improvements

Version 0.6.1

Veröffentlicht am 20.5.2015

- Compatibility with Debian jessie

Version 0.6

Veröffentlicht am 7.9.2013

- Bugfix: Die Datei `/opt/zpub/demo/cache/documents` spiegelt jetzt stets den neusten Revisionsstand wieder; nicht den der zuletzt gebauten Dokumentrevision (welcher älter sein kann).
- In der Datei `/opt/zpub/demo/conf/logo.png` kann ein Logo hinterlegt werden. (Dazu muss die Apache-Konfiguration angepasst werden.)
- Bugfix: Dateien, die im Grundverzeichnis des SVN-Repositories angelegt werden, stoßen keine Dokumenten-Bau-Jobs an.
- Es gibt jetzt eine Testsuite für zpub.
- Der Pfad `/opt/zpub/demo/output/archive/`*`document`*`/latest` ist jetzt ein Verzeichnis, das pro Stil einen symbolischen Link auf die letzte Revision mit diesem Stil enthält.
- Es ist jetzt möglich, im Verzeichnis `common` Dateien (Bilder und Dokumentfragmente) direkt in den Dokumenten zu verwenden. zpub erkennt bei Änderungen an diesen Dateien, welche Dokumente von der Änderung betroffen sind und baut nur jene Dokumente neu. *Dieses Feature wurde per Crowdfunding [http://www.zpub.de/de/crowdfunding.html] finanziert*:

  ```
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" \
      href="../common/fragment.xml" />
  ```

- Bei der Erstellung der HTML-Ausgabe werden wirklich nur verwendete Bilder kopiert, und die Dokumenterstellung schlägt fehl, wenn nicht alle referenzierten Bilder gefunden werden können.

Version 0.5

7.11.2012

- Dokumente, die nicht mehr im Subversion-Repository liegen, werden auch nicht mehr in der Web-Oberfläche angezeigt.
- Die zpub-Dokumentation wird jetzt mit installiert und kann mit **zpub-update-docs.sh** in eine Instanz eingefügt bzw. darin aktualisiert werden. Dabei wird in der Dokumentations der zur Instanz gehörende Instanz- und Hostname eingefügt.
- Diese Versionshistorie.

Version 0.4

Veröffentlicht am 10.2.2012

- Ab dieser Version kann man auch mehrere zpub-Instanzen auf dem selben virtuellen Host unterbringen, da eine Instanz auch in einem Unterverzeichnis des URL-Adressraumes eingerichtet werden kann.
- Es können mehrere Ausgabevarianten („styles“) gleichzeitig konfiguriert werden.

Version 0.3

Veröffentlicht am 2.1.2012

- Das Ausgabeformat epub für mobile Lesegeräte wird unterstützt.
- Die aktuelle Version der zpub-Installation wird im Web-Interface angezeigt.

Version 0.2

Veröffentlicht am 6.12.2010

- Es gibt jetzt ein Installationsskript und eine Paketierung für Debian- und Ubuntu-Pakete.
- Die generierten Ausgabeformate sind jetzt konfigurierbar.
- Ein Backup der gesamten Dokumentquellen samt Historie kann über das Web-Interface heruntergeladen werden.

Systemvoraussetzungen

zpub ist, technisch gesehen, eine Sammlung von Bash- und Perl-Skripten, die im Zusammenspiel mit apache2 und subversion die erwartete Funktionalität bereitstellen. Dazu müssen folgende Pakete eines Debian-Systems, Release “Lenny”, installiert sein:

apache2	cabextract	docbook-xsl	fop
libapache2-svn	libdatetime-format-strptime-perl-perl	libdatetime-perl	libfile-slurp-perl
libfilesys-df-perl	libipc-run-perl	libmime-lite-perl	libpaper1
libsaxon-java	libsvn-svnlook-perl	libsys-cpuload-perl	libtemplate-perl
realpath	rsync	subversion	sun-java6-jre
wine	xsltproc	zip	

Weiter muss für die Erstellung von Windows-Hilfedateien der Microsoft HTML Help Workshop installiert sein. Dazu führt man als der Systembenutzer, der später die Hilfedateien erstellen wird, folgende Befehle aus:

```
$ wget http://htmlhelp.googlecode.com/svn/trunk/misc/htmlhelp.sh
$ wget http://htmlhelp.googlecode.com/svn/trunk/misc/htmlhelp.reg
$ bash ./htmlhelp.sh
```

Für die korrekte Wortrennung von deutschen Texten muss von der Webseite http://sourceforge.net/projects/offo das Paket offo-hyphenation-fop-stable heruntergeladen und die darin enthaltene Datei `fop-hyph.jar` im Verzeichnis `/opt/zpub/tools` gespeichert werden.

zpub - für Debian paketiert

zpub ist als Debian-Paket verfügbar, welches die Installation und Ersteinrichtung stark vereinfacht bzw. automatisiert. Ein Installationsskript für zpub-Instanzen (`zpub-create-instance`) ist ebenfalls vorhanden.

Einrichtung über das zpub-Debian-Paket:

1. Erweiterung der `/etc/apt/sources.list` um das private zpub-Repository:

   ```
   deb http://zpub.de/debian/ ./
   ```

2. `$ apt-get install zpub`:

   - neuer Benutzer "zpub" wird erstellt und der Gruppe "`www-data`" hinzugefügt
   - ein vorhandener Apache-Webserver wird für zpub konfiguriert (ein neuer vhost wird erstellt)

3. ```
   $ zpub-create-instance
   Usage:

   /usr/sbin/zpub-create-instance name 'Full Name' zpub.domain.com

   where
    name:            Directory name of the instance in /var/lib/zpub
    'Full Name':     Name as shown in the web interface
    zpub.domain.com: Hostname for this virtual host
   ```
   ```

```
$ zpub-create-instance testzpublocal "test-zpub auf Localhost"  zpub.localhos
```

Die neue zpub-Instanz "`testzpublocal`" ist nun angelegt.

Weitere Schritte sind beispielsweise:

- `style`-Verzeichnis anpassen bzw. eigene Stylesheets hinterlegen
- via `htpasswd` neue zpub-Benutzer anlegen
- zpub-admin Benutzer anlegen
- eventuell `/etc/hosts` anpassen
- Apache neu starten, um die neue zpub-Instanz zu erreichen

4. zpub ist über den Browser erreichbar:

   bspw.: `https://zpub.localhost/` (Hostname wie bei `zpub-create-instance` (s.o.) angegeben)

5. zpub-Benutzer anlegen (hier: `test`):

   ```
   htpasswd -b /var/lib/zpub/testzpublocal/settings/htpasswd test test
   ```

6. Datenverzeichnis anlegen
7. Datenverzeichnis mit Dokument-Verzeichnis(sen) füllen
8. svn: neues Dokumentverzeichnis dem SVN-Repository hinzufügen (`svn add`), Änderungen übertragen (`svn ci`)
9. zpub-Dokumente werden generiert und sind über das Webinterface, welche unter der oben definierten Domain läuft, abrufbar

Anmerkung

- `/var/lib/zpub/`→ enthält ein Verzeichnis je instanz
- `/var/lib/zpub/testzpublocal/conf/formats`→ Aktivierung/Deaktivierung der gewünschten Ausgabeformate
- `/var/lib/zpub/testzpublocal/conf/default_style`→ Standard-Stil (z.B. `draft`)
- `/var/lib/zpub/testzpublocal/conf/final_style`→ Alternativ-Stil (z.b. `final`)

Verzeichnisstruktur

zpub erwartet eine bestimme Verzeichnisstruktur für seine Daten, seine Konfiguration, die Skripte und die Ausgabe. Es geht davon aus, dass all dies in `/opt/zpub` liegt. Neben den allgemeinen Verzeichnissen darin gibt es auch ein Verzeichnis pro Kunde, im folgenden exemplarisch `/opt/zpub/demo` benannt.

zpub-Verzeichnisse

`/opt/zpub/bin`	ausführbare Skripte, etwa der SVN-Post-Commit-Hook, das DocBook-Render-Skript oder das CGI-Skript.

`/opt/zpub/bin/lib`	gemeinsam genutzte Programmteile, etwa Perl-Module
`/opt/zpub/templates`	Template-Toolkit-Vorlagen für das Web-Frontend und die Benachrichtigungs-e-Mail
`/opt/zpub/templates/static`	Statische Dateien, etwa CSS-Style-Sheets, die unter `/static` in das Web-Interface eingebunden werden.
`/opt/zpub/tools`	Werkzeuge, die weder selbst entwickelt wurden (und demnach `/opt/zpub/bin` liegen), noch als Debian-Paket zur Verfügungen stehen; etwa der Worttrenner für fop.
`/opt/zpub/spool`	In diesem Verzeichnis werden die Render-Aufträge verwaltet. Es gibt die folgenden Unterverzeichnisse: In `new/` werden neue Aufträge, etwa nach einem SVN-Commit, angelegt und dann nach `todo/` verschoben. Der Spooler findet sie dort und schiebt sie nach `wip/` (für „work in process“), während sie bearbeitet und gelöst werden. Im Fehlerfalle werden sie nach `fail/` verschoben.
`/opt/zpub/repos/zpub`	enthält das SVN-Repository für alle nicht-benutzerspezifischen, selbst entwickelten Dateien. Für den produktiven Betrieb wird dieses nicht benötigt.
`/opt/zpub/demo`	Verzeichnisbaum für alle benutzer-spezifischen Dateien, hier am Beispiel demo.
`/opt/zpub/demo/conf`	Konfigurationsdateien, die der Benutzer nicht selbst bearbeiten darf. Enthält neben der Apache-Konfiguration `apache2.conf` auch verschiedene Dateien, die das zpub-Verhalten selbst steuern. Diese sind weiter unten, in der Liste „zpub-Einstellungen“ erklärt.
`/opt/zpub/demo/settings`	Konfigurationsdateien, die der Benutzer selbst einstellen darf, etwa per Web-Frontend. Enthält neben der Apache-Benutzerdatenbank `htpasswd` auch weitere Dateien, die das zpub-Verhalten steuern. Diese sind weiter unten, in der Liste „zpub-Einstellungen“ erklärt.
`/opt/zpub/demo/style`	Verfügbare Stylesheets, eines pro Unterverzeichnis, deren Inhalt in „Aufbau der Stile“ erläutert werden.
`/opt/zpub/demo/repos/style`	Ein SVN-Repository, dass das oben genannte Verzeichnis verwaltet. Für den produktiven Betrieb wird dieses nicht benötigt.
`/opt/zpub/demo/repos/source`	Das SVN-Repository mit den eigentlichen XML-Quellen der Dokumente. Im Repository existiert für jedes Dokument ein Verzeichnis mit *genau einer* XML-Datei. Weitere Dateien, wie etwa Bilder, können in Unterverzeichnissen gespeichert sein. Für den produktiven Betrieb wird dieses Repository direkt verwendet, es gibt keine dauerhaft ausgecheckte Arbeitskopie.
`/opt/zpub/demo/output`	Hier werden die generierten Dokumente abgelegt. Es gibt ein Unterverzeichnis pro Dokument, das wiederum das Verzeichnis `archive/` enthält. Darin wird für jede verfügbare Kombination aus SVN-Revisionsnummer und XSL-Stil ein Verzeichnis angelegt. Diese können gefahrlos gelöscht werden, wenn Festplattenspeicher frei gemacht werden soll.

Auch gibt es, neben dem Verzeichnis `archive/`, pro Dokument ein Verzeichnis `latest`. Dieses enthält pro Stil einen symbolischen Link auf die jeweils neuste Revision des Stils Er kann verwendet werden, um geeignete Bookmarks zu setzen.

`/opt/zpub/demo/cache` — In diesem Verzeichnis legt zpub diverse interne Dateien ab. Diese müssen nicht von Hand bearbeitet werden, dürfen aber auch nicht gelöscht werden.

Wie bereits oben erwähnt wird zpub über die Dateien in den Verzeichnissen `/opt/zpub/demo/conf` und `/opt/zpub/demo/settings` gesteuert. Diese sind meist einfache Text-Dateien, die nur eine Wert speichern, seltener eine Liste von Werten, die dann in jeweils einer Zeile stehen.

Manche Einstellungen, wie etwa `/opt/zpub/demo/settings/final_rev/`, sind dokumenten-spezifisch und sind selbst Verzeichnisse, die pro Dokument je eine wie das Dokument benannte Datei enthalten.

zpub-Einstellungen

`conf/cust_name` — Der Name des Kunden, in menschenlesbarer Schreibweise.

`conf/logo.png` — Ein Kunden-spezifisches Logo im PNG-Format, Auflösung 136×42 Pixel.

`conf/hostname` — (Optional) Der Name des virtuellen Hosts, standardmäßig demo.zpub.de.

`conf/rootpath` — (Optional) Der Pfad der URL der zpub-Instanz, wenn etwa mehrere zpub-Instanzen auf einem virtuellen Host laufen sollen. Standardmäßig leehr, das heißt dass zpub im Wurzelverzeichnis der Website läuft. Die Variable sollte mit einem Slash beginnen, aber ohne Slash enden, etwa „`/zpub-instance`“. Diese Einstellung ist unabhängig von den Pfaden der Dateien im Dateisystem!

`conf/admins` — eine zeilenweise Aufzählung der Benutzer, die in der Web-Oberfläche Admin-Rechte haben.

`conf/features` — Eine zeilenweise Aufzählung aller für diesen Kunden aktivieren Features. Momentan definierte Features sind:

- `final_approve`: Ein Admin-Benutzter kann eine Revision eines Dokuments als final freigeben, dieses wird dann mit einem speziellen Stil erzeugt (etwa ohne den Vermerkt „draft“).
- `online_backup`: Ein Admin-Benutzter kann auf der Status-Seite das komplette Dokumenten-Quellarchiv als SVN-Dump herunterladen.

`conf/default_style` — das standardmäßig zu verwendende XSLT-Stylesheet, spezifiziert über den Verzeichnisnamen in `/opt/zpub/demo/style/`.

`conf/final_style` — falls das Feature `final_approve` aktiv ist: Welcher Stil für freigegebene Dokumente verwendet werden soll.

`conf/formats` — enthält eine Zeile pro Format (`html`, `pdf` oder `pdfhelp`), in dem das Dokument erstellt werden soll. (Ab version 0.2)

`settings/final_rev/` — falls das Feature `final_approve` aktiv ist: Welche Revision des Dokuments die aktuell freigegebene ist. (dokumenten-spezifisch)

`settings/subscribers/` — Eine komma-separierte Liste der bei Änderungen am Dokument zu benachrichtigenden Benutzer. (dokumenten-spezifisch)

Beispiel:

```
Max Mustermann <max.mustermann@web.de>,
Petra Musterfrau <petra@musterfrau.de>
```

Zusammenspiel der Komponenten

Die Skripte im Verzeichnis `/opt/zpub/bin` führen jeweils spezifische Ausgaben aus, so dass im Zusammenspiel die gewünschte Funktionalität geboten wird. In diesem Kapitel wird der Programmablauf für verschiedene Anwendungsfälle beschreiben.

Erstellung einer neuen Dokumenten-Revsion

Dieser Anwendungsfall beginnt auf dem Rechner des Redakteurs, der ein vorhandenes Dokument lokal bearbeitet oder ein neues anlegt. Mit seinem Subversion-Client überträgt er seine Änderung auf den zpub-Server. Nun gibt es eine neue Revision des Dokuments, die über die von SVN vergebenen Nummer identifiziert wird.

SVN startet direkt nach dem Commit das Skript `zpub-post-commit-hook.sh`. Dieses untersucht die soeben gemachten Änderungen um herauszufinden, welche Dokumente bearbeitet wurden. Es ist natürlich möglich, mit einem SVN-Commit mehrere Dokumente zu ändern. Für jedes geänderte Dokumement legt es einen Auftrag in `/opt/zpub/spool/todo` an, welches Angaben zu

- Kunde
- Dokument
- Revision
- zu verwendenter Stil
- und Ausgabeverzeichnis

enthält. Dabei wird der zu verwendente Stil aus `conf/default_style` gelesen, und das Ausgabeverzeichnis in `/opt/zpub/demo/output` nach dem oben beschriebenen Schema erstellt.

Außerdem aktualisiert `zpub-post-commit-hook.sh` die Datei `/opt/zpub/demo/cache/documents`, die die aktuell vorhandenen Dokumente aufzählt. Wird ein Dokument im SVN-Repository gelöscht, wird es darauf hin auch nicht mehr angezeigt. Die generierten Dokumente sind aber weiterhin vorhanden und müssen ggf. vom Systemadministrator gelöscht werden.

Das Skript `zpub-spooler.sh`, welches permanent läuft, schaut regelmäßig in `/opt/zpub/spool/todo` nach neuen Aufträgen. Diese verschiebt es nacheinander nach `/opt/zpub/spool/wip` und ruft das Skript `zpub-render.sh` mit den Parametern aus der Auftrgs-Datei auf.

Das Render-Skript erstellt darauf das Ausgabeverzeichnis, falls es nicht schon existiert. Es extrahiert die gewünschte Revision des Dokumentes aus dem SVN-Repository, legt eine symbolische Verknüpfung zu dem gewählten Dokumenten-Stil an und ruft, entsprechend der Konfiugration von `conf/formats`, die Skripte `zpub-render-`*`format`*`.sh`, welche die einzelnen Ausgabeformate bauen. Dabei kommt der Microsoft HTML Help Workshop zum Einsatz, sowie die Werkzeuge xslt und fop. Bei jeder Art von Fehler bricht dieses Skript ab.

Je nach dem, ob das Render-Skript erfolgreich war oder nicht, wird die Auftragsdatei vom Spooler gelöscht oder nach `/opt/zpub/spool/fail` verschoben. Der Spooler ruft dann auch noch das

Skript `zpub-send-mail.pl` auf, welches gegebenenfalls an die in `settings/subscribers` aufgeführten e-Mail-Adressen eine Benachrichtigung über die neue Revision schickt, sowie das Skript `zpub-link-latest.pl`, welches den oben erwähnten symbolischen Link anlegt. Danach wartet der Spooler wieder auf neue Aufträge.

Aufbau der Stile

Das Verzeichnis `/opt/zpub/demo/style` enthält ein Verzeichnis pro verfügbarem Stil (hier Beispielshaft `/opt/zpub/demo/style/plain`, welches die XSLT-Dateien und zugehörigen Mediendateien enhalten. Folgende Dateien im Verzeichnis `plain` haben besondere Bedeutung:

Stil-Dateien

`pdf/fo.xsl`	XSLT-Stylesheet für die PDF-Ausgabe
`pdf/fop.xconf`	fop-Konfigurationsdatei (optional)
`html/html.xsl`	XSLT-Stylesheet für die HTML-Ausgabe
`html/static/`	Zusätzliche Dateien für die HTML-Ausgabe
`htmlhelp/htmlhelp.xsl`	XSLT-Stylesheet für die Windows-Hilfe-Ausgabe
`htmlhelp/static/`	Zusätzliche Dateien für die Windows-Hilfe-Ausgabe
`epub/epub.xsl`	XSLT-Stylesheet für die epub-Ausgabe
`epub/static/`	Zusätzliche Dateien für die epub-Ausgabe

Ein Bild `tip.png`, das im Wurzelverzeichnis des Style-Verzeichnisses liegt, muss wie folgt in den Stylesheets referenziert werden:

- Stylesheets für HTML, HtmlHelp und epub verwenden den Pfad `images/../style/tip.png`.
- Stylesheets für PDF verwenden den Pfad `style/tip.png`.

Beim erstellen der Zip-Archive für HTML, epub und Windows-Hilfe werden nur jene Bilder mit aufgenommen, die in den Dokumenten auch referenziert werden. Die Bilddateien oder die Unterverzeichnisse, in denen sie liegen, dürfen dabei auch symbolische Links sein. So könnte etwa `xsl-images` ein symbolischer Link auf `/usr/share/xml/docbook/stylesheet/docbook-xsl/images/` sein.

Zusätzliche Dateien wie CSS-Stile und JavaScript-Dateien werden nicht automatisch kopiert; auch nicht darin referenzierte Dateien. Diese müssen daher in dem jeweiligen `static/`-Unterverzeichnis bereitgestellt werden. Eine Datei, die etwa im HTML-Stylesheet über einen Pfad `css/screen.css.` referenziert wird, muss dazu in `html/static/css/screen.css` liegen. Auch müssen Bilder, die via CSS oder JavaScript geladen werden, im Verzeichnis `static/` liegen.

Übersetzung

Anmerkung

In zpub ist bisher kein Übersetzungsworkflow integriert. Übersetzungen können jedoch im ausgecheckten zpub-Repository mit Hilfe der Software `xml2po` vorgenommen werden. Dieser Abschnitt behandelt den Übersetzungsworkflow anhand des Beispieldokuments `handbuch_de/zpub-Technik.xml`.

Beispiel 1. Repository Verzeichnisstruktur vor der Übersetzung

```
zpub-repository/
|-- handbuch_de
|   |-- i18n
|   `-- zpub-Technik.xml
`-- manual_en
```

1. `$ apt-get install xml2po`
2. Erzeugen der `pot`-Template Datei:

   ```
   $ xml2po -o handbuch_de/i18n/zpub-Technik.pot \
       handbuch_de/zpub-Technik.xml
   ```

3. Die erzeugte Datei `zpub-Technik.pot` wird übersetzt. Eventuell hilfreich sind Software-Lösungen, die das Übersetzen unterstützen (z.B. poedit.net [https://poedit.net/]). Die Übersetzung wird als `LANG.po`-Datei im Verzeichnis `i18n` gespeichert (z.B.: `i18n/en_GB.po`).
4. Aus dem Quelldokument `zpub-Technik.xml` und der Übersetzungsdatei `en_GB.po` wird das (übersetzte) DocBook-Dokument erzeugt:

   ```
   $ xml2po -p handbuch_de/i18n/en_GB.po \
       -o manual_en/zpub-manual.xml \
       handbuch_de/zpub-Technik.xml
   ```

5. Das neue zpub-Dokument `zpub-manual.xml` wird via SVN commited und zpub generiert die definierten Ausgabeformate für das übersetzte Dokument.

Übersetzung aktualisieren

Werden am Ursprungsdokument inhaltliche Änderungen vorgenommen, muss die Übersetzungsdatei `LANG.po` aktualisiert werden:

```
$ xml2po -u handbuch_de/i18n/en_GB.po handbuch_de/zpub-Technik.xml
```

Die neuen/geänderten Übersetzungen werden wie oben in Merge po-Datei eingearbeitet und das zu übersetzende Dokument neu generiert (siehe Ausgabeformate generieren).

Beispiel 2. Repository Verzeichnisstruktur nach der Übersetzung

```
zpub-repository/
|-- handbuch_de
|   |-- i18n
|   |   |-- en_GB.po
|   |   `-- zpub-Technik.pot
|   `-- zpub-Technik.xml
`-- manual_en
    `-- zpub-manual.xml
```

Authentifizierung und Authentisierung

Authentifizierung und Authentisierung erfolgen primär über die vorgestellte zpub-eigene Benutzerverwaltung. Es ist jedoch auch möglich, die in zpub verwendeten Komponenten (hier: apache und subversion) z.B. gegen einen LDAP-Verzeichnisdienst authentifizieren zu lassen oder Single Sign-On einzurichten.

LDAP

Auf dem Verzeichnisdienst muss ein Abfrage-User (hier: `apacheconnect`) eingerichtet sein.

Aapche-Module:
a2enmod authnz_ldap ldap

Beispiel 3. /etc/apache2/mods-enabled/ldap.conf

```
LDAPVerifyServerCert Off|On
```

Beispiel 4. vhost.conf (Ausschnitt)

```
AuthType Basic
AuthBasicProvider ldap file
AuthUserFile /var/lib/zpub/${instance-name}/settings/htpasswd
AuthLDAPURL ldaps://dc1.domain.tld:636/dc=domain,dc=tld\
    ?sAMAccountName?sub?(objectClass=*)
AuthLDAPBindDN apacheconnect@domain.tld
AuthLDAPBindPassword geheim
Require ldap-group ou=zpubgruppe,DC=domain,DC=tld
```

Die LDAP-Abfragen können mit **ldapsearch** aus dem Debian-Paket 'ldap-utils' getestet werden:

```
$ ldapsearch -x -D "apacheconnect@domain.tld" -w geheim -b "dc=domain,dc=tld" \
    -H "ldaps://dc1.domain.tld:636" "(sn=apacheconnect)" dn
```

Single sign-on (Kerberos)

Anmerkung

Die Zeit muss auf allen beteiligten Hosts korrekt gestellt sein. Weiter ist auf eine korrekte Namensauflösung (DNS-Lookup wie auch Reverse-DNS-Lookup) sowie auf die korrekte Klein- und Großschreibung für die Kerberos-Konfiguration zu achten.

Anmerkung

Alle Angaben beziehen sich auf die Realisierung der Authentifizierung und Authentisierung gegen ein Microsoft Active Directory™ bzw. folgende Umgebung:

- Domain Controller: Windows Server 2012R2™
- HTTP Service: Apache 2.4/Debian 8.x
- Client: Microsoft Windows 7™

Anmerkung

Bei der Nutzung von Kerberos ist keine "fallback"-Authentifizierung durch andere Authentifizierungs-Provider (z.B. `file`) möglich.

1. Host AD: Erstellen eines Service Principals im Active Directory: Neuer Benutzer anlegen

Anzeigename	Kerberos Service Principal
Benutzeranmeldename	`HTTP/$zpub-instance-name.domain.tld`
Kontooption	Kennwort läuft nie ab
Kontooption	Dieses Konto unterstützt Kerberos-AES-256-Bit-Verschlüsselung

2. Host AD: `keytab` für diesen Service Principal erstellen

```
C:\>ktpass -princ HTTP/$zpub-instance-name.domain.tld@DOMAIN.TLD \
```

```
-mapuser kerberosprinc@DOMAIN.TLD \
-crypto AES256-SHA1 -ptype KRB5_NT_PRINCIPAL \
-pass not_be_guessable -out c:\keyfile
```

Anmerkung

Der Name Service Principal, sprich der Teil zwischen dem `HTTP/` und dem `@DOMAIN.TLD`, muss exakt auf den FQDN des zu "kerberisierenden" Hosts lauten.

3. Host zpub: `keytab`-Datei auf zpub-Host übertragen und Dateisystemberechtigungen anpassen:

```
$ chown www-data http_zpub.krb5keytab
$ chmod  400 http_zpub.krb5keytab
```

4. Host zpub: LDAP-Konfiguration

`/etc/ldap/ldap.conf`:

```
BASE    dc=domain,dc=tld
URI     ldap://dc1.domain.tld
```

5. Host zpub: Kerberos-Konfiguration

`/etc/krb5.conf`:

```
[libdefaults]
    default_realm = DOMAIN.TLD

[realms]
    DOMAIN.TLD = {
        kdc = dc1.domain.tld
        admin_server = dc1.domain.tld
        default_domain = domain.tld
     }

[domain_realm]
    .domain.tld = DOMAIN.TLD
    domain.tld = DOMAIN.TLD
```

6. Host zpub: Testen von Ticket und Keytab-Datei:

```
$ kinit -k -t http_zpub.krb5keytab  HTTP/$zpub-instance-name.domain.tld
$ kvno  HTTP/$zpub-instance-name.domain.tld@DOMAIN.TLD
$ klist -e
$ klist -e -k -t http_zpub.krb5keytab
```

Drei Angaben müssen jeweils übereinstimmen:

a. `kvno`

b. `principal name`

c. `encryption type`

Die zum Testen erstellen Tickets sollten wieder gelöscht werden:

```
$ kdestroy
```

7. Host zpub: apache-Module aktivieren:

```
$ a2enmod auth_kerb ldap authnz_ldap
```

8. Host zpub: apache konfigurieren:

```
AuthName "zpub single Sign On"
AuthType Kerberos
KrbAuthRealms DOMAIN.TLD
KrbServiceName HTTP/${zpub-instance-hostname}.domain.tld@DOMAIN.TLD
Krb5Keytab /etc/apache2/http_${zpub-instance-hostname}.krb5keytab
KrbMethodNegotiate on
KrbMethodK5Passwd off
KrbAuthoritative off
KrbLocalUserMapping on
#KrbVerifyKDC off
#KrbSaveCredentials on

AuthBasicProvider ldap
AuthLDAPURL ldap://dc1.domain.tld:3268/dc=domain,dc=tld?\
    sAMAccountName?sub?(objectClass=*)
AuthLDAPBindDN apacheconnect@DOMAIN.TLD
AuthLDAPBindPassword not_be_guessable

<RequireAll>
    Require ldap-group CN=zpub,DC=domain,DC=tld
</RequireAll>
```

9. Client: Webbrowser für SSO konfigurieren:

   Firefox: `network.negotiate-auth.trusted-uris`: domain.tld

   IE: Host zu `Trusted Sites` hinzufügen