

independIT Integrative Technologies GmbH
Bergstraße 6
D-86529 Schrobenhausen



schedulix

Installationshandbuch Release 2.11

Dieter Stubler
Ronald Jeninga

Copyright © 2026 independIT GmbH

Rechtlicher Hinweis

Dieses Werk ist urheberrechtlich geschützt.

Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Buches, oder Teilen daraus, vorbehalten. Kein Teil des Werkes darf ohne schriftliche Genehmigung der independIT GmbH in irgendeiner Form (Fotokopie, Mikrofilm oder ein anderes Verfahren), auch nicht für Zwecke der Unterrichtsgestaltung, reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltsverzeichnis

Inhaltsverzeichnis	1
1 Voraussetzungen	3
Compile Umgebung	3
schedulix Server	3
schedulix Client	4
Zope Application Server	5
2 Compile des Systems	7
Generelle Vorbereitung	7
Compile	7
3 Installation in einer Linux-Umgebung	9
Installation des schedulix Servers	9
Installation eines schedulix Clients	12
Beispiel Installation eines Jobserver	14
Szenario	14
Vorraussetzungen	14
Installation	14
Installation mit Postgres	16
Einleitung	16
Installation	16
Installation mit MySQL	17
Einleitung	17
Installation	17
Installation mit Ingres	18
Einleitung	18
Installation	18
Konfiguration von TLS/SSL Verbindungen	20
Einleitung	20
Konfiguration von TLS/SSL	20
Installation (Zope4+)	23
Migration einer bestehenden BICsuite!Web Zope2 Datenbank nach Zope4+	26
HTTPS mit Hilfe eines vorgeschaltetem Apache Servers	26
SSO für schedulix mit Zope	27
Einleitung	27
Vorgehen	28
Kerberos Installation und Konfiguration	28
Apache Webserver und Module	30
Zope Erweiterung und Konfiguration	33
Konfiguration des schedulix Servers	39
Einstellungen an Benutzerseite	40
Administration des Zope Servers	41

Kapitel 1

Voraussetzungen

Compile Umgebung

Um aus dem Source Paket auf einem Linux System die benötigten Executables zu erstellen, wird folgende Software benötigt:

- Oracle(Sun) Java 1.8 JDK oder höher
<http://www.oracle.com/technetwork/java/index.html>
Alternativ dazu ein OpenJDK 1.8 oder höher
<http://openjdk.java.net>
- gcc, gcc-c++
<http://gcc.gnu.org>
- gnu make
<http://www.gnu.org/software/make>
- jflex (Version 1.4.x)
<http://jflex.de>
- jay
Das jay Executable wird zwar im Paket mitgeliefert, aber einen Hinweis auf die Originalquellen bzw. -executables sollte hier nicht fehlen.
<http://www.cs.rit.edu/~ats/projects/lp/doc/jay/package-summary.html>
- Eclipse SWT
schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Damit der Compile nicht abbricht wird ein Eclipse-SWT benötigt.
<http://www.eclipse.org/swt>
- Java Native Access (JNA)
Um die Notwendigkeit einer JNI Library zu umgehen wird ab der Version 2.6 die JNA Bibliothek genutzt.
<https://github.com/twall/jna>

In vielen Fällen können die benötigte Software Pakete einfach über ein Package Manager wie yum, rpm oder dpkg installiert werden.

schedulix Server

Zur Installation des schedulix Servers wird folgende Software benötigt:

- Ein in einer Compile Umgebung für dieselbe Zielarchitektur erzeugtes schedulix-2.11.tgz
- Oracle(Sun) Java 1.8 SE jre
<http://www.oracle.com/technetwork/java/index.html>
 Alternativ dazu ein OpenJDK 1.8 oder höher
<http://openjdk.java.net>
- Eines der folgenden RDBMS Systeme mit zugehörigem JDBC Interface
 - PostgreSQL
<http://www.postgresql.org>
 JDBC für PostgreSQL:
<http://jdbc.postgresql.org>
 - MySQL
<http://www.mysql.com>
 MySQL (Connector/J) JDBC Driver
<http://www.mysql.com>
 - Ingres
<http://www.ingres.com>
- Eclipse SWT
 schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Sollen diese Beispiele installiert werden, wird ein Eclipse-SWT benötigt.
<http://www.eclipse.org/swt>
 Andernfalls ist ein SWT nicht erforderlich.

schedulix Client

Zur Installation eines schedulix Clients wird folgende Software benötigt:

- Ein in einer Compile Umgebung für dieselbe Zielarchitektur erzeugtes schedulix-2.11.tgz
- Oracle(Sun) Java 1.8 SE jre
<http://www.oracle.com/technetwork/java/index.html>
 Alternativ dazu ein OpenJDK 1.8 oder höher
<http://openjdk.java.net>
- Eclipse SWT
 schedulix enthält einige Beispiele, die ein installiertes SWT voraussetzen. Sollen diese Beispiele auch auf dem Client ausgeführt werden können, wird ein Eclipse-SWT benötigt.
<http://www.eclipse.org/swt>
 Andernfalls ist ein SWT nicht erforderlich.
- Java Native Access (JNA)
 Um die Notwendigkeit einer JNI Library zu umgehen wird ab der Version 2.6 die JNA Bibliothek genutzt. Diese Bibliothek wird nur für den Jobserver benötigt.
<https://github.com/twall/jna>

Zope Application Server

Das Web Frontend wird von dem Zope Application Server bereitgestellt. Zur Installation des Zope Servers wird folgende Software benötigt:

- Python 2.7
<http://www.python.org>
- Python development package (python-devel oder python-dev)
<http://www.python.org>
- python-setuptools
<http://pypi.python.org>

Kapitel 2

Compile des Systems

Generelle Vorbereitung

Es ist sinnvoll eine Software mit zentraler Bedeutung unter einem eigenen Account zu installieren. Dies vereinfacht die Administration und schützt gegen Missbrauch. Im nachfolgenden wird davon ausgegangen, dass die Umwandlung und Installation unter dem Account `schedulix` erfolgt. Als Home-Verzeichnis wird `/home/schedulix` angenommen. Selbstverständlich sind dies alles nur Vorschläge. Es gibt keine technische Notwendigkeit diese Vorschläge an zu nehmen. Allerdings muss die Anleitung bei Änderungen entsprechend interpretiert werden.

Wie ein Benutzer angelegt werden kann steht, im Installationskapitel auf Seite 9 dokumentiert.

Compile

Um nach der Installation der benötigten Pakete das System erfolgreich zu übersetzen, müssen noch einige Umgebungsvariablen gesetzt werden bevor "make" die eigentlichen Arbeit machen kann.

Da weder für das Umwandeln als auch für die Installation an sich keine besondere Rechte benötigt werden, wird unter dem User `schedulix` gearbeitet.

1. Download der `schedulix` Source Distribution von github
Alle zur Übersetzung und Installation notwendigen Dateien stehen in einem github Repository zur Verfügung und können mit folgendem Kommando heruntergeladen werden:

```
cd $HOME
git clone https://github.com/schedulix/schedulix.git
-b v2.11 schedulix-2.11
```

Danach befinden sich alle Dateien der `schedulix` Source Distribution im Unterverzeichnis:

```
$HOME/schedulix-2.11
```

2. Setzen der Umgebungsvariablen
Nun müssen einige Umgebungsvariablen gesetzt werden. Die folgende Kommandos stammen von einer Installation auf einer CentOS Linux Distribution (<http://www.centos.org>). In vielen Fällen können die Befehle eins-zu-eins übernommen werden, aber sie sind abhängig von der genauen Linux Distribution und installierten Software.

```
export SDMSHOME=/home/schedulix/schedulix-2.11
export CLASSPATH=$CLASSPATH:/usr/share/java/jflex.jar
export JAVAHOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.0
export SWTJAR=/usr/lib/java/swt.jar
export JNAJAR=/usr/share/java/jna.jar
```

Es empfiehlt sich diese Einstellungen in die `.bashrc` auf zu nehmen, zumindest so lange das Umwandeln des Systems noch nicht erledigt ist.

3. make

Es bleibt jetzt nur noch das tatsächliche Umwandeln des Systems übrig.

```
cd ~/schedulix-2.11/src
make
```

Bei einem wiederholten Versuch das System um zu wandeln, empfiehlt es sich statt `make make new` ein zu geben.

Als letzte Aktion wird ein jar-File erzeugt und unter `/schedulix-2.11/lib` abgelegt.

4. Erzeugen `/schedulix-2.11.tgz`

```
cd $HOME
tar czf schedulix-2.11.tgz schedulix-2.11
```

Kapitel 3

Installation in einer Linux-Umgebung

Installation des schedulix Servers

Die Installation des schedulix Scheduling Servers ist einfach. Es bedarf nur einiger Handlungen die im Folgenden erläutert werden:

Wenn (Beispiel-)Kommandos vorgestellt werden, wird als Prompt normalerweise ein `$` gezeigt. Diese Kommandos werden dann unter dem neu anzulegenden Account `schedulix` ausgeführt. In einigen Fällen wird der privilegierte Account `root` benötigt. Dies wird dadurch gekennzeichnet, dass als Prompt ein `#` gezeigt wird.

5. User `schedulix` anlegen

Es gibt keine Notwendigkeit den User `schedulix` zu nennen. Damit kann der Name auch einer beliebigen Konvention angepasst werden. In diesem Dokument wird davon ausgegangen, dass der User `schedulix` heißt.

Unter Ubuntu Linux kann ein User folgendermaßen angelegt werden:

```
# useradd -d /home/schedulix -m -s /bin/bash -U schedulix
# passwd schedulix
```

Alle nachfolgenden Aktionen werden unter User `schedulix` ausgeführt, es sei denn es wird explizit anders angegeben.

6. Herunterladen und Installieren eines von schedulix unterstützten Datenbank Management Systems.

`schedulix` für Linux unterstützt derzeit die Systeme:

- Postgres (Seite [16](#))
- MySQL (Seite [17](#))
- Ingres (Seite [18](#))

Für die Installation des gewählten Datenbanksystems, sowie die Anpassung der Konfiguration des schedulix Enterprise Scheduling Systems, wird auf die entsprechenden nachfolgenden Abschnitte verwiesen.

7. Software auspacken

tar-Archiv auspacken im Homeverzeichnis von schedulix . Etwa:

```
$ tar xvzf schedulix-2.11.tgz
```

Symbolic Link anlegen:

```
$ ln -s schedulix-2.11 schedulix
```

8. Konfiguration erstellen

(a) Benutzerumgebung

Um mit dem schedulix System arbeiten zu können, müssen folgende Variablen gesetzt werden:

```
BICSUITEHOME=/home/schedulix/schedulix
BICSUITECONFIG=/home/schedulix/etc
PATH=$BICSUITEHOME/bin:$PATH
SWTJAR=/usr/lib/java/swt.jar
JNAJAR=/usr/share/java/jna.jar
```

Es hat sich in der Praxis als vorteilhaft erwiesen die Konfiguration des Systems außerhalb des Installationsverzeichnisses zu legen. Damit werden spätere Upgrades wesentlich erleichtert. Da die Variablen von allen Benutzern des Systems gesetzt werden müssen, kann es sinnvoll sein die Zuweisungen (und Exports) in einer eigenen Datei zu schreiben, und diese dann im `.profile` oder `.bashrc` zu sourcen.

(b) Softwareumgebung

Unter `$BICSUITEHOME/etc` liegen einige Vorlagen für Konfigurationsdateien, die gut als Basis für die eigene Systemkonfiguration verwendet werden können. Diese müssen dazu ohne die Endung `".template"` in das Konfigurationsverzeichnis `$BICSUITECONFIG` kopiert werden.

Etwa

```
$ cd $BICSUITEHOME/etc; for fff in *.template; do
> TRG=`basename $fff .template`;
> cp $fff $BICSUITECONFIG/$TRG;
> done
```

Anschließend müssen die Dateien natürlich der Umgebung angepasst werden.

Die Datei `bicsuite.conf` setzt einige Default-Einstellungen und muss im Allgemeinen nicht angepasst werden. Allerdings kann man sich überlegen das Logging des Systems außerhalb des Installationsverzeichnisses stattfinden zu lassen. Dazu muss lediglich die Variable `BICSUITELOGDIR` entsprechend angepasst werden. Das in `BICSUITELOGDIR` gesetzte Verzeichnis muss vorhanden sein.

Die Datei `java.conf` beschreibt die zu verwendende Java-Umgebung. Insbesondere muss der Pfad zum JDBC-Treiber eingegeben werden. Weiterhin wird die Speicherkonfiguration des Servers geregelt. Dazu muss, auch in großen Umgebungen, normalerweise nur die Variable `BICSUITEMEM` angepasst werden.

Die Datei `server.conf` enthält die Serverkonfiguration. Angepasst werden müssen hier die Einstellungen für die Verbindung des schedulix Scheduling Serves zu seinem RDBMS Repository. Mehr dazu finden Sie im jeweiligen Kapitel zum eingesetzten RDBMS.

Weiterhin muss in dieser Datei das Property `hostname` auf den Hostnamen oder die IP-Adresse des Servers gesetzt werden.

Die Datei `jobserver.conf` wird hier nicht benötigt, dient aber als Vorlage für die Jobserver-Konfiguration.

9. Datenbank einrichten

Abhängig davon welches Datenbanksystem Sie nutzen möchten, befolgen Sie die Anleitung zur Einrichtung der Datenbank.

Für

- Ingres, siehe Seite 18,
- MySQL, siehe Seite 17, und für
- PostgreSQL, siehe Seite 16.

10. Server hochfahren

Die Installation ist nun im Wesentlichen abgeschlossen. Was noch bleibt ist das Starten des Servers und, bei Bedarf, das Einspielen der Beispiele.

Der Server kann mittels

```
$ server-start
```

gestartet werden.

11. Anlegen der Datei `.sdmshrc`

Die Datei `.sdmshrc` wird, falls vorhanden, von allen `schedulix` Kommandozeilen-Werkzeugen gelesen um Kommandozeilen-Parameter vorzubelegen. Im Folgenden wird davon ausgegangen, dass diese Datei existiert und für User, Passwort, Host und Port die korrekten Werte gesetzt enthält. Die Datei `.sdmshrc` wird im Home-Verzeichnis des Linux-Benutzers angelegt.

Ein Beispiel für den Inhalt ist:

```
$ cat ~/.sdmshrc
User=SYSTEM
Password=G0H0ME
Host=localhost
Port=2506
Timeout=0
```

Wichtig: Da die Datei die Daten für den Zugang zum Scheduling Server enthält, sollten die Datei-Rechte so gesetzt sein, dass nur der Owner die Datei lesen kann.

```
$ chmod 600 ~/.sdmshrc
$ ls -lG ~/.sdmshrc
-rw----- 1 schedulix 73 2011-11-09 09:28 /home/schedulix/.sdmshrc
```

12. Convenience Package installieren

Das Convenience Package installiert eine übliche Konfiguration eines Exit State Modells.

```
$ sdmsh < $BICSUITEHOME/install/convenience.sdms
```

13. Beispiele installieren (optional)

Das Installieren der Beispiele besteht aus zwei Teilen: Zum einen werden drei sogenannte Jobserver angelegt, welche für die nachfolgenden Ablaufdefinitionen benötigt werden. Zum anderen werden Beispiele für Ablaufdefinitionen in den Server geladen.

(a) Anlegen der Jobserver

Um die Jobserver anzulegen, muss nur ein Skript ausgeführt werden:

```
$ cd $BICSUITEHOME/install
$ setup_example_jobserver.sh
```

(b) Einspielen der Ablaufdefinitionen

Zum Einspielen der Ablaufdefinitionen werden folgende Befehle eingegeben:

```
$ cd $BICSUITEHOME/install
$ sdmsh < setup_examples.sdmsh
```

Da die Beispiele davon ausgehen, dass die Jobserver bereits angelegt wurden, ist die obige Reihenfolge **zwingend**.

Installation eines schedulix Clients

Die Installation eines schedulix Scheduling Clients ist einfach. Es bedarf nur einiger Handlungen die im Folgenden erläutert werden.

Wenn (Beispiel-)Kommandos vorgestellt werden, wird als Prompt normalerweise ein `$` gezeigt. Diese Kommandos werden dann unter dem neu anzulegenden Account `schedulix` ausgeführt. In einigen Fällen wird der privilegierte Account `root` benötigt. Dies wird dadurch gekennzeichnet, dass als Prompt ein `#` gezeigt wird.

5. User schedulix anlegen

Es gibt keine Notwendigkeit den User `schedulix` zu nennen. Damit kann der Name auch einer beliebigen Konvention angepasst werden. In diesem Dokument wird davon ausgegangen, dass der User `schedulix` heißt.

Unter Ubuntu Linux kann ein User folgendermaßen angelegt werden:

```
# useradd -d /home/schedulix -m -s /bin/bash -U schedulix
# passwd schedulix
```

Alle nachfolgenden Aktionen werden unter User `schedulix` ausgeführt, es sei denn es wird explizit anders angegeben.

6. Software auspacken

tar-Archiv auspacken im Homeverzeichnis von `schedulix`. Etwa:

```
$ tar xvf schedulix-2.11.tgz
```

Symbolic Link anlegen:

```
$ ln -s schedulix-2.11 schedulix
```

7. Konfiguration erstellen

(a) Benutzerumgebung

Um mit dem schedulix System arbeiten zu können, müssen folgende Variablen gesetzt werden:

```
BICSUITEHOME=/home/schedulix/schedulix
BICSUITECONFIG=/home/schedulix/etc
PATH=$BICSUITEHOME/bin:$PATH
SWTJAR=/usr/lib/java/swt.jar
JNAJAR=/usr/share/java/jna.jar
```

Es hat sich in der Praxis als vorteilhaft erwiesen die Konfiguration des Systems außerhalb des Installationsverzeichnis zu legen. Damit werden spätere Upgrades wesentlich erleichtert. Da die Variablen von allen Benutzern des Systems gesetzt werden müssen, kann es sinnvoll sein die Zuweisungen (und Exports) in einer eigenen Datei zu schreiben, und diese dann im `.profile` oder `.bashrc` zu sourcen.

(b) Softwareumgebung

Unter `$BICSUITEHOME/etc` liegen einige Vorlagen für Konfigurationsdateien, die als Basis für die Systemkonfiguration verwendet werden sollten.

Für eine Client-Installation werden die Dateien `bicsuite.conf` und `java.conf` benötigt.

Diese müssen dazu ohne die Endung `".template"` ins Konfigurationsverzeichnis `$BICSUITECONFIG` kopiert werden.

```
$ cp $BICSUITEHOME/etc/bicsuite.conf.template \
    $BICSUITECONFIG/bicsuite.conf
$ cp $BICSUITEHOME/etc/java.conf.template \
    $BICSUITECONFIG/java.conf
```

Die Datei `bicsuite.conf` setzt einige Default-Einstellungen und muss im Allgemeinen nicht angepasst werden.

Die Datei `java.conf` beschreibt die zu verwendende Java-Umgebung und muss im Allgemeinen nicht weiter angepasst werden.

8. Anlegen der Datei `.sdmshrc`

Die Datei `.sdmshrc` wird, falls vorhanden, von allen schedulix Kommandozeilen-Werkzeugen gelesen um Kommandozeilen-Parameter vorzubelegen. Im Folgenden wird davon ausgegangen, dass diese Datei existiert und für User, Passwort, Host und Port die korrekten Werte gesetzt enthält. Die Datei `.sdmshrc` wird im Home-Verzeichnis des Linux-Benutzers angelegt.

Ein Beispiel für den Inhalt ist:

```
$ cat ~/.sdmshrc
User=SYSTEM
Password=G0H0ME
Host=localhost
Port=2506
Timeout=0
```

Wichtig: Da die Datei die Daten für den Zugang zum Scheduling Server enthält, sollten die Datei-Rechte so gesetzt sein, dass nur der Owner die Datei lesen kann.

```
$ chmod 600 ~/.sdmshrc
$ ls -lG ~/.sdmshrc
-rw----- 1 schedulix 73 2011-11-09 09:28 /home/schedulix/.sdmshrc
```

Beispiel Installation eines Jobserver

Im Folgenden wird die Installation eines Jobserver anhand eines Beispiels durchgeführt.

Szenario

Auf dem Rechner `marvin` soll ein Jobserver Prozesse als Benutzer `arthur` ausführen. Das home Verzeichnis des Benutzers sei `/home/arthur`.

Der `schedulix` Server sei auf dem Rechner `schedmain` installiert und hört auf den Port 2506. Das System-Passwort sei `G0H0ME`.

Vorraussetzungen

Auf dem Rechner `marvin` wurde eine `schedulix` Client-Installation im home Verzeichnis `/home/schedulix` durchgeführt.

Der Benutzer `arthur` benötigt folgende Zugriffsberechtigungen auf die Dateien der Client-Installation:

- Leserechte auf die Dateien `java.conf` und `bicsuite.conf` im Konfigurationsverzeichnis `$BICSUITECONFIG`, sowie auf alle Dateien unter `/home/schedulix/lib`.
- Lese- und Ausführungsrechte auf alle Dateien unter `/home/schedulix/bin`.

Installation

Damit ein Jobserver sich am `schedulix` Scheduling Server anmelden kann muss der Jobserver dem `schedulix` Scheduling Server bekannt gemacht und konfiguriert werden. Im Folgenden führen wir die notwendigen Schritte dazu auf Kommandozeilenebene mit dem Werkzeug `sdmsh` durch. Dies kann jedoch alternativ auch über das Web GUI getan werden.

1. Anmelden als Benutzer `arthur` auf dem Rechner `marvin`
2. Setzen der Umgebungsvariablen in der Shell und `.bashrc`.

```
export BICSUITEHOME=/home/schedulix/schedulix
export BICSUITECONFIG=/home/schedulix/etc
export PATH=$BICSUITEHOME/bin:$PATH
```

3. Testen, ob die Umgebung korrekt ist

```
sdmsh --host localhost --port 2506 --user SYSTEM --pass G0H0ME
```

Ein `SDMS>` Prompt sollte erscheinen ('exit' beendet `sdmsh`).

4. Anlegen der Verzeichnisse

```
cd $HOME
mkdir etc
mkdir taskfiles
mkdir work
mkdir log
```

5. Erzeugen einen Scopes für die Maschine `marvin` mit `sdmsh`

```
SDMS> CREATE OR ALTER SCOPE GLOBAL.'MARVIN'
WITH
CONFIG = (
  'JOBEXECUTOR' = '/home/schedulix/schedulix/bin/jobserver',
  'HTTPHOST' = 'marvin'
);
```


Die Pfade müssen explizit angegeben werden, die Benutzung von Umgebungsvariablen ist hier nicht möglich.

6. Erzeugen des Jobserver mit sdmsh

```
SDMS> CREATE OR ALTER JOB SERVER GLOBAL.'MARVIN'.'ARTHUR'
WITH
  PASSWORD = 'dent',
  NODE = 'marvin',
  CONFIG = (
    'JOBFILEPREFIX' = '/home/arthur/taskfiles/',
    'DEFAULTWORKDIR' = '/home/arthur/work',
    'HTTPPORT' = '8905',
    'NAME_PATTERN_LOGFILES' = '/home/arthur/work/.*\\.log'
  );
```

Der HTTPPORT ist für die Anzeige von Job Log-Dateien aus dem Browser notwendig und ist beliebig wählbar, muss aber für alle Jobserver auf derselben Maschine eindeutig sein.

Die Pfade müssen explizit angegeben werden, die Benutzung von Umgebungsvariablen ist hier nicht möglich.

7. Anlegen der Named Resource für den Jobserver mit sdmsh

```
SDMS> CREATE OR ALTER NAMED RESOURCE RESOURCE.'JOBSERVERS'
WITH USAGE = CATEGORY;
SDMS> CREATE NAMED RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MARVIN'
WITH USAGE = STATIC;
```

8. Anlegen eines Environments für den Jobserver mit sdmsh

```
SDMS> CREATE ENVIRONMENT 'ARTHUR@MARVIN'
WITH RESOURCE = (RESOURCE.'JOBSERVERS'.'ARTHUR@MARVIN');
```

9. Anlegen der Resource im Jobserver mit sdmsh

```
SDMS> CREATE RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MARVIN'
IN GLOBAL.'MARVIN'.'ARTHUR' WITH ONLINE;
```

10. Erzeugen der Konfigurationsdatei \$HOME/etc/jobserver.conf für den Jobserver mit folgendem Inhalt:

```
RepoHost= schedmain
RepoPort= 2506
RepoUser= "GLOBAL.'MARVIN'.'ARTHUR'"
RepoPass= dent
```

11. Starten des Jobserver

```
jobserver-run $HOME/etc/jobserver.conf $HOME/log/jobserver.out
```

Soll der Jobserver automatisch mit dem Start des Rechners gestartet werden, ist dies vom Systemadministrator entsprechend einzurichten.

Installation mit Postgres

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die Postgres-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine "Standard" Installation durchzuführen.

Installation

1. Herunterladen und Installieren der aktuellen Postgres-Version

Normalerweise wird für jede Linux Distribution ein Postgres Package angeboten. Dieses Package, sowie ein Package für den JDBC Treiber für Postgres, sollte problemlos installiert werden können.

2. Konfiguration der Datei `pg_hba.conf`

Damit sich der `schedulix` Scheduling Server mit Benutzer und Passwort bei PostgreSQL authentifizieren kann, muss folgende Zeile in die Postgres-Konfigurationsdatei `pg_hba.conf` aufgenommen werden. Diese Datei liegt typischerweise im Verzeichnis `/var/lib/pgsql/<version>/data`.

```
host          all          all          127.0.0.1/32          md5
```

PostgreSQL muss dann neu gestartet werden.

3. Anlegen des Postgres Users `schedulix`

Führen Sie als User `postgres` den Befehl `createuser` wie im Beispiel (Version 8) aus:

```
$ createuser -P schedulix
Enter password for new role:
Enter it again:
Shall the new role be a superuser? (y/n): n
Shall the new role be allowed to create databases? (y/n): y
Shall the new role be allowed to create more new roles? (y/n): n
```

beziehungsweise, für Version 9:

```
$ createuser -P -d schedulix
```

Das eingegebene Passwort wird später noch benötigt.

4. Anlegen der Repository Datenbank `schedulixdb`

Legen Sie nun als Benutzer `schedulix` die Datenbank für das Repository wie im untenstehenden Beispiel an:

```
$ createdb schedulixdb
```

5. Anlegen und Initialisierung der Datenbanktabellen

Um das benötigte Datenbankschema anzulegen, wechseln Sie in das `schedulix` SQL-Verzeichnis und rufen Sie das Postgres Utility `psql` wie im untenstehenden Beispiel auf:

```
$ cd $BICSUITEHOME/sql
$ psql -f pg/install.sql schedulixdb
```

6. Konfigurieren der Datenbankverbindung in der schedulix Server Konfigurationsdatei `$BICSUITECONFIG/server.conf`

Ändern Sie folgende Properties wie angegeben:

```
DbPasswd=schedulix passwort
DbUrl=jdbc:postgresql:schedulixdb
DbUser=schedulix
JdbcDriver=org.postgresql.Driver
```

Die `DbUrl` ist etwas abhängig von der installierten PostgreSQL-Version. Unter Version 8 lautet sie

```
DbUrl=jdbc:postgresql:schedulixdb
```

7. Konfigurieren Sie den schedulix Java Class Path für Postgres JDBC

In der Konfigurationsdatei `$BICSUITECONFIG/java.conf` muss nun nur noch der Pfad zum Postgres JDBC-Treiber am `CLASSPATH` angehängt werden.

Etwa

```
BICSUITECLASSPATH=$BICSUITEJAR:/usr/share/java/postgresql-jdbc4-9.2.jar
```

Installation mit MySQL

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die MySQL-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine "Standard" Installation durchzuführen.

Installation

1. Herunterladen und Installieren der aktuellen MySQL-Version.

Für die meisten Linux-Distributionen gibt es fertige MySQL Packages. Diese können mit den entsprechenden Tools einfach installiert werden.

Im Rahmen dieser Installation wird nach einem Passwort für den MySQL root-User gefragt (nicht zu verwechseln mit dem Linux root-User). Dieses Passwort wird im nächsten Schritt wieder benötigt.

Da schedulix für den Zugriff auf die Datenbank eine JDBC Connection aufbaut, muss auch der MySQL JDBC-Treiber installiert werden.

2. Anlegen des MySQL Users `schedulix` und der Datenbank `schedulixdb`

Starten Sie das Utility `mysql` und melden Sie sich als MySQL root-User an um den User `schedulix` sowie die Datenbank `schedulixdb` anzulegen:

```
$ mysql --user=root --password=mysql-root-password
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.1.54-lubuntu4 (Ubuntu)
```

```
Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql> create user schedulix identified by 'schedulix_passwort';
Query OK, 0 rows affected (0.01 sec)

mysql> create database schedulixdb;
Query OK, 1 row affected (0.00 sec)

mysql> grant all on schedulixdb.* to schedulix;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

3. Anlegen und Initialisierung der Datenbanktabellen

Führen Sie folgende Kommandos aus:

```
$ cd $BICSUITEHOME/sql
$ mysql --user=schedulix --password=schedulix_passwort
  --database=schedulixdb --execute="source mysql/install.sql"
```

4. Konfigurieren der Datenbankverbindung in der schedulix Server-Konfigurationsdatei

`$BICSUITECONFIG/server.conf`

Ändern Sie folgende Properties wie angegeben:

```
DbPasswd=schedulix_passwort
DbUrl=jdbc:mysql:///schedulixdb
DbUser=schedulix
JdbcDriver=com.mysql.jdbc.Driver
```

5. Konfigurieren Sie den schedulix Java Class Path für MySQL JDBC

In der Konfigurationsdatei `$BICSUITECONFIG/java.conf` muss nun nur noch der Pfad zum MySQL JDBC-Treiber an dem `CLASSPATH` angehängt werden.

Etwas

```
BICSUITECLASSPATH=$BICSUITEJAR:/usr/share/java/mysql-connector-java.jar
```

Installation mit Ingres

Einleitung

Diese Anleitung erhebt nicht den Anspruch eine genaue Beschreibung der Installation des Datenbanksystems zu sein. Dazu wird auf die Ingres-Dokumentation verwiesen. Im Normalfall sollte es mit dieser Anleitung allerdings möglich sein eine "Standard"-Installation durchzuführen.

Installation

1. Installation von Ingres

Wir gehen davon aus, dass das Ingres-System unter User `ingres` installiert wird. Der Installations-Identifizier wird hier als `II`, was dem Standardwert entspricht, angenommen.

2. Anlegen des Users `schedulix`

Um den Benutzer `schedulix` im Ingres-System bekannt zu machen, gibt es zwei Möglichkeiten. Als Erste kann der Benutzer mit Hilfe des Tools `accessdb` angelegt werden. Diese Möglichkeit wird hier nicht weiter erläutert.

Die zweite Möglichkeit ist das Anlegen des Benutzers mittels SQL-Befehl. Dazu starten Sie als Ingres den SQL Terminal Monitor:

```

$ su - ingres
Password:
ingres@cheetah:~$ sql iidbdb
INGRES TERMINAL MONITOR Copyright 2008 Ingres Corporation
Ingres Linux Version II 9.2.1 (a64.lnx/103)NPTL login
Mon Jun 13 10:05:19 2011

continue
* create user schedulix with privileges = (createdb);
* \g
Executing . . .

continue
* commit;\g
Executing . . .

continue
* \q
Ingres Version II 9.2.1 (a64.lnx/103)NPTL logout
Mon Jun 13 10:07:58 2011
ingres@cheetah:~$

```

3. Anlegen der Repository Datenbank schedulixdb

```

$ $II_SYSTEM/ingres/bin/createdb schedulixdb
Creating database 'schedulixdb' . . .

Creating DBMS System Catalogs . . .
Modifying DBMS System Catalogs . . .
Creating Standard Catalog Interface . . .
Creating Front-end System Catalogs . . .

Creation of database 'schedulixdb' completed successfully.

```

4. Anlegen und Initialisierung der Datenbanktabellen

Zum Anlegen der benötigten Tabellen führen Sie folgende Befehle durch:

```

$ cd $BICSUITEHOME/sql
$ sql schedulixdb < ing\install.sql

```

5. Konfigurieren der Datenbankverbindung in der schedulix Server Konfigurationsdatei \$BICSUITECONFIG/server.conf

Ändern Sie folgende Properties wie angegeben:

```

DbPasswd=<schedulix OS password>
DbUrl=jdbc:ingres://localhost:II7/schedulixdb;
DbUser=schedulix
JdbcDriver=com.ingres.jdbc.IngresDriver

```

6. Konfigurieren Sie den schedulix Java Class Path für Ingres JDBC

In der Konfigurationsdatei \$BICSUITECONFIG/java.conf muss nun nur noch der Pfad zum Ingres JDBC Treiber an dem CLASSPATH angehängt werden.

Etwa

```

BICSUITECLASSPATH=$BICSUITEJAR:$II_SYSTEM/ingres/lib/iijdbc.jar

```

Konfiguration von TLS/SSL Verbindungen

Einleitung

Das Aufsetzen einer verschlüsselten Kommunikation innerhalb des schedulix Systems ist relativ einfach. Allerdings wird der Aufwand, je nach Anforderungen, variieren.

Es gibt folgende Möglichkeiten:

1. schedulix ohne SSL/TLS Kommunikation
2. schedulix mit und ohne SSL/TLS Kommunikation
3. schedulix mit ausschließlich SSL/TLS Kommunikation

Wenn mittels TLS/SSL kommuniziert wird, gibt es wiederum zwei Möglichkeiten:

1. Nur serverseitige Authentifizierung: Das bedeutet, dass Clients überprüfen können, ob der Server mit dem sie kommunizieren, auch tatsächlich vertrauenswürdig ist. Diese Einstellung erfordert nur ein Keypair für den Server. Jegliche Kommunikation erfolgt verschlüsselt.
2. Server- und clientseitige Authentifizierung: Bei dieser Konfiguration wird von beiden Seiten geprüft, ob die Identität des Kommunikationspartners bekannt ist. Diese Einstellung ist sehr sicher, aber aufwendig da für jeden Client und natürlich für den Server ein Keypair erzeugt werden muss. Selbstverständlich erfolgt die Kommunikation verschlüsselt.

Konfiguration von TLS/SSL

Das Aufsetzen der SSL/TLS Kommunikation erfordert im Wesentlichen einige wenige Aktionen.

1. Generieren des Keypairs für den Server

Zum Generieren der Keypairs wird das Utility `keytool`, was Bestandteil von Java (SE) ist, genutzt. Nähere Information zu diesem Utility gibt es z.B. unter

<https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html>

Folgendes Vorgehen sollte funktionieren (für "nicht"-Spezialisten):

- (a) Legen Sie ein Verzeichnis an, z.B.

```
$ mkdir $BICSUITECONFIG/certs
```

- (b) Das Verzeichnis wird nachher den private Key des Servers enthalten und sollte somit entsprechend geschützt werden, z.B.

```
$ chmod 700 $BICSUITECONFIG/certs
```

- (c) Generieren Sie ein Keypair mit Hilfe von `keytool`, z.B.

```
$ keytool -genkeypair -alias schedulix -keypass secret \  
> -dname "cn=servername, ou=schedulix, o=independIT, c=DE" \  
> -keystore $BICSUITECONFIG/certs/svrkeystore \  
> -storepass secret -validity 365
```

WICHTIG: Die beiden Passwörter (`keypass` und `storepass`) *müssen* gleich sein!

- (d) Der public Key vom Server muss später noch den Clients mitgeteilt werden. Dazu muss dieser zuerst aus dem Keyfile extrahiert werden.

```
$ keytool -export -alias schedulix -file svrkey \  
> -keystore $BICSUITECONFIG/certs/svrkeystore
```

Es wird dazu das `storepass` (`secret`) benötigt.

2. Anpassen der Serverkonfiguration

Die Serverkonfiguration muss nun angepasst werden um dem Server die Information über seine Keys zukommen zu lassen. Weiterhin muss dabei entschieden werden, ob nur eine serverseitige Authentifizierung oder eine beidseitige Authentifizierung stattfinden soll. Auch muss eine Entscheidung bezüglich der unverschlüsselten Kommunikation getroffen werden.

```
#
# SSLPort: der Port fuer die verschluesselte Kommunikation
# (inaktiv = 0)
#
SSLPort=2507
#
# KeyStore definiert welche Datei mit Keys genutzt werden soll
# $BICSUITEHOME/etc/certs/avrkeystore
#
KeyStore=/home/schedulix/etc/certs/avrkeystore
#
# Das KeyStorePassword wird benoetigt um den Keystore zu lesen
#
KeyStorePassword=secret
#
# TrustStore definiert welche Datei die (public) Keys der
# gueltigen Kommunikationspartner enthaelt
#
TrustStore=/home/schedulix/etc/certs/avrkeystore
#
# Das TrustStorePassword wird benoetigt um den Truststore zu lesen
#
TrustStorePassword=secret
#
# ClientAuthentication besagt ob Clients sich ebenfalls
# autorisieren muessen (true), oder nicht (false)
#
ClientAuthentication=true
#
# Port definiert den Port fuer die unverschluesselte Kommunikation
# (inaktiv = 0)
# Wenn alle Ports als inaktiv konfiguriert sind,
# wird Port 2506 fuer unverschluesselte Kommunikation geoeffnet
#
Port=2506
#
# ServicePort definiert den Port fuer System Zugang in Notfaellen
# (inaktiv = 0)
#
ServicePort=2505
```

Wenn Client Authentication=true ist, werden vom Server sowohl der Keystore als auch der Truststore benötigt. Wenn eine Client-Authentifizierung nicht erforderlich ist, wird nur der Keystore benötigt.

3. Client Konfiguration

Wenn keine Client-Authentifizierung erforderlich ist, benötigen Clients nur den Zugriff auf ein Truststore um die Identität des Servers überprüfen zu können. Muss auch die Identität der Clients überprüft werden, muss für jeden Client auch ein Keystore erzeugt werden. Dies erfolgt analog zu der Erzeugung des Keystores für den Server.

4. Anpassen von .sdmshrc

Wenn sdmsh bzw. die Standard Utilities über TLS/SSL mit dem Server kommunizieren soll ist die Benutzung eines "ini"-Files zwingend erforderlich. Es gibt drei Möglichkeiten:

- (a) \$BICSUITECONFIG/sdmshrc
- (b) \$HOME/.sdmshrc
- (c) Eine beim Aufruf des Utilities spezifizierte Datei

Entscheidend dabei ist die Angabe der für den Aufbau der sicheren Verbindung benötigten Information. Etwa

```
User=donald
Password=duck
Host=localhost
Port=2507
SSL=true
KeyStore=/home/schedulix/etc/certs/clntkeystore
TrustStore=/home/schedulix/etc/certs/clnttruststore
KeyStorePassword=secret
TrustStorePassword=secret
Timeout=0
```

was eine symmetrische Authentifizierung erlaubt. Ist nur eine serverseitige Authentifizierung erforderlich, können die Keystore betreffenden Zeilen entfernt werden.

5. Jobserver Konfiguration

Die Jobserver haben vier neue Konfigurationsparameter entsprechend den Serverparametern bekommen:

```
KEYSTORE
TRUSTSTORE
KEYSTOREPASSWORD
TRUSTSTOREPASSWORD
```

Dazu kommt noch ein Parameter der angibt, ob eine verschlüsselte Kommunikation erwünscht ist:

```
USE_SSL
```

Wenn Client Authentication erwünscht ist, muss für jeden Jobserver ein Schlüssel-paar angefertigt werden. Dies geht genauso wie für den schedulix Server beschrieben wurde.

Natürlich muss darauf geachtet werden, dass insbesondere der Konfigurationsparameter RepoPort richtig gesetzt ist.

6. Getting it all together

Last but not least müssen nun die public Keys zwischen den einzelnen Kommunikationspartnern ausgetauscht werden. Auch dies erfolgt mit dem Utility keytool. Um etwa den public Key des Servers als vertrauenswürdig zu definieren, wird er in den Truststore des Clients aufgenommen. z.B.:

```
$ keytool -import -keystore clntkeystore -alias schedulix -file svrkey
Enter keystore password:
Owner: CN=servername, OU=schedulix, O=independIT, C=DE
Issuer: CN=servername, OU=schedulix, O=independIT, C=DE
Serial number: 4dc28814
Valid from: Thu May 05 13:20:52 2011 until: Fri May 04 13:20:52 2012
Certificate fingerprints:
    MD5: D3:83:F2:2B:93:2B:65:7A:41:3E:CE:2E:C8:EC:40:62
    SHA1: AF:B1:18:95:B2:2A:BB:1D:08:BD:A6:87:68:64:6B:FC:0D:A8:30:DA
    Signature algorithm name: SHA1withDSA
    Version: 3
Trust this certificate? [no]: yes
Certificate was added to keystore
```


Dazu wird natürlich wiederum ein Storepass benötigt.

Wird auch die Client-Authentifizierung verlangt, müssen auch die Client Certificates in den Truststore des Servers eingetragen werden.

Installation (Zope4+)

1. Voraussetzungen

Um Zope4 bzw. Zope5 zu installieren müssen folgende Pakete installiert werden:

(a) python3

```
$ sudo yum install python3
```

(b) python3 development headers

```
$ sudo yum install python3-devel
```

Selbstverständlich kann, je nach Distribution, auch ein Tool wie apt oder Vergleichbares benutzt werden.

2. Erzeugen der virtuellen Python-Umgebung für die Zope-Installation

```
$ export INSTALLDIR=$HOME/software
$ export ZOPE5VENV=Zope5
$ export ZOPE5DIR=$INSTALLDIR/$ZOPE5VENV
$ mkdir -p $INSTALLDIR
$ cd $INSTALLDIR
$ python3 -m venv $ZOPE5VENV
```

3. Installieren der Zope5 Software

Installieren Sie die neueste Release von Zope5. Zum Zeitpunkt der Erstellung dieser Dokumentation war Zope 5.1.2 die aktuellen Version.

```
$ cd $ZOPE5DIR
$ bin/pip install -U pip wheel
$ bin/pip install Zope[wsgi]==5.0 \
-c https://zopefoundation.github.io/Zope/releases/5.1.2/constraints.txt
$ bin/pip install Products.ExternalMethod
$ bin/pip install Products.Sessions
$ bin/pip install Products.SiteErrorLog
$ bin/pip install Products.PythonScripts
$ bin/pip install requests
```

Kann bei der Installation nicht auf das Internet zugegriffen werden, kann Zope auch offline installiert werden. Dazu ist wie folgt vorzugehen:

(a) Download python packages

Auf einem möglichst identischen System mit Internetzugang führen Sie folgende Kommandos aus:

```
$ wget \
https://zopefoundation.github.io/Zope/releases/5.9/requirements-full.txt
$ pip download -r requirements.txt -d packages
```

(b) Dateien auf das Zielsystem übertragen

Die Datei requirements.txt und das Verzeichnis packages müssen nun auf das Zielsystem ohne Internetzugang übertragen werden. Legen Sie die Dateien in \$HOME/software ab.

(c) Installation auf dem Zielsystem

Folgendes Kommando installiert Zope aus den heruntergeladenen Dateien:

```
$ cd $HOME/software
$ Zope/bin/pip install --no-index --use-wheel \
--find-links=./packages -r requirements.txt
```

4. Erzeugen einer Zope-Instanz für schedulix !Web

```
$ cd $HOME
$ export ZOPE5INSTANCE=$HOME/Zope5Instance
$ $ZOPE5DIR/bin/mkwsgiinstance -d $ZOPE5INSTANCE \
-u sdmsadm:sdmsadm_passwort
```

Das Passwort kann beliebig gewählt werden und wird später wieder benötigt. Der Benutzer muss aber `sdmsadm` heißen.

Test:

Starten Sie die Zope5 Instanz mit folgendem Befehl:

```
$ $ZOPE5DIR/bin/runwsgi -v $ZOPE5INSTANCE/etc/zope.ini
```

Im Browser sollte die URL <http://localhost:8080> nun eine 'Zope Auto-generated default page' zeigen.

Zope5 kann nun mit Strg-C oder durch Schließen der Windows-Eingabeaufforderung wieder beendet werden.

5. Installieren der schedulix !Web-Komponenten

Um die schedulix !Web-Komponenten zu installieren, muss die Zope-Installation um einige Module erweitert werden:

```
$ cd $ZOPE5INSTANCE
$ mkdir Extensions
$ cd Extensions
$ ln -s $BICSUITEHOME/zope4/Extensions/*.py .
$ cd ..
$ ln -s $BICSUITEHOME/zope4/Products/BICsuiteSubmitMemory \
$ZOPE5DIR/lib64/python3*/site-packages/Products
$ ln -s $BICSUITEHOME/zope4/Products/StringFixer \
$ZOPE5DIR/lib64/python3*/site-packages/Products
$ mkdir import
$ cd import
$ ln -s $BICSUITEHOME/zope4/import/SDMS.zexp .
```

Nun muss die Zope-Instanz wieder gestartet werden, um die Änderungen auch Zope-seitig bekannt zu machen.

```
$ $ZOPE5DIR/bin/runwsgi -v $ZOPE5INSTANCE/etc/zope.ini
```

Die Zope Management Oberfläche wird nun unter der Adresse

<http://localhost:8080/manage>

mit Hilfe eines Browsers geöffnet. Dazu wird der Benutzer `sdmsadm` mit dem von Ihnen vergebenen Passwort benutzt.

Es wird jetzt die Frontend Software in Zope geladen (Import Button)

(a) im Folder / SDMS.zexp importieren

(b) im Folder /SDMS/Install die Folder User und Custom anwählen und mit Copy kopieren

(c) im Folder / mit Paste die Folder User und Custom erzeugen

6. Serververbindungen konfigurieren

Das Konfigurieren der Serververbindungen erfolgt ebenfalls aus der Zope Management-Oberfläche heraus. Dazu meldet man sich als Benutzer `sdmsadm` an.

Im Folder Custom wird das PythonScript `SDMSServers` editiert. Dieses Skript liefert ein Dictionary, welches für jeden `schedulix` Server, der von dieser `schedulix !Web` Installation angesprochen werden soll, einen Eintrag der Form

```
# Servername unter dem der Server in der schedulix!web Oberflaeche
# sichtbar ist
'servername' : {
    # IP Adresse oder Hostname auf dem der schedulix!server laeuft
    'HOST'      : 'hostname',

    # Port unter dem der schedulix!server angesprochen wird
    'PORT'      : '2506',

    # BASIC
    'VERSION'   : 'BASIC',

    # optionales Property, ob Zope Serververbindungen cachen soll
    'CACHE'     : 'Y'

    # optionales Property, wie lange gecachte schedulix!web
    # Serververbindungen gueltig sein sollen
    # default ist 60 sekunden, nur von Bedeutung falls 'CACHE' : 'Y'
    'TIMEOUT'   : '60'
}
```

enthalten muss. Fürs Bootstrapping muss ein Eintrag mit Namen `DEFAULT` vorhanden sein. Dieser Eintrag kann nach dem Einrichten der Benutzer (die dann diese Connection natürlich nicht benutzen sollten) entfernt werden.

Soll ein Server über eine sichere SSL-Verbindung angesprochen werden, dann müssen folgende weitere Eigenschaften definiert werden:

```
# Verbindung wird ueber Secure Socket Layer aufgebaut
'SSL'      : 'true',

# falls angegeben, wird die Identitaet des BICsuite Servers
# ueberprueft. Die angegebene Datei muss das Server Certificate
# des BICsuite Servers enthalten
'TRUSTSTORE' : 'truststore.pem',

# falls der BICsuite Server eine Client Authentication fordert,
# muss dieses Property definiert sein und die angegebene Datei
# muss das Certificate und den Private Key des Clients enthalten.
# Das Certificate muss dem Server in seinem Truststore bekannt sein.
'KEYSTORE'  : 'keystore.pem'
```

Anmerkung:

Bei Verwendung von SSL wird aus Performancegründen die Verwendung von cached Serververbindungen empfohlen, da der Aufbau einer gesicherten Verbindung eine rechenintensive Operation ist.

7. Die `schedulix !Web` Oberfläche öffnen

Die Benutzeroberfläche steht nun unter der Adresse

`http://localhost:8080/SDMS`

bereit. Nach dem Öffnen dieser Seite erscheint eine Aufforderung zur Anmeldung. Nach der Anmeldung wird die Applikation dann mit dem "Take Off" Button gestartet.

Für das weitere Arbeiten mit der Oberfläche sei nun auf die dazugehörige Dokumentation verwiesen.

Migration einer bestehenden BICsuite!Web Zope2 Datenbank nach Zope4+

Um die Datenbank einer bestehenden BICsuite!Web Zope2 Installation in eine neue BICsuite!Web Zope4+ Installation zu übernehmen, sind folgende Schritte durchzuführen:

1. Führen Sie alle Schritte der vorangegangenen Anleitung zur "Installation (Zope4+)" durch.
2. Stoppen Sie die installierte BICsuite!web Zope4+ Instanz falls diese aktiv ist
3. Installieren zodbupdate

Für die Migration der bestehenden BICsuite!web Zope2 Datenbank nach Zope4+ muss das tool zodbupdate installiert werden.

```
$ cd $ZOE5DIR
$ bin/pip install zodbupdate
```

4. Stoppen Sie Ihre BICsuite!web Zope2 Instanz falls diese aktiv ist
5. Übernehmen Sie das Data.fs aus Ihrer BICsuite!web Zope2 Instanz in Ihre Zope4+ Instanz

Setzen dafür die Umgebungsvariablen ZOPE5DIR, ZOPE2INSTANCE und ZOPE5INSTANCE auf die für Sie gültigen Verzeichnisse.

```
$ rm $ZOPE5INSTANCE/var/Data.fs*
$ cp $ZOPE2INSTANCE/var/Data.fs $ZOPE5INSTANCE/var
$ $ZOPE5DIR/bin/zodbupdate -v -f var/Data.fs --convert-py3 --encoding utf8
```

6. Starten Sie Ihre BICsuite!web Zope4+ Instanz
7. Ersetzen des SDMS Folders
 - (a) Öffnen Sie die Management Oberfläche Ihrer Zope4+ Instanz
 - (b) Lösche Sie den temp_folder aus dem Root (/) Folder
 - (c) Navigieren Sie zu dem Folder, welcher den Folder SDMS enthält
 - (d) Löschen Sie den Folder SDMS
 - (e) Importieren Sie das SDMS.zexp

HTTPS mit Hilfe eines vorgeschaltetem Apache Servers

Die einfachste Möglichkeit das HTTPS Protokoll anstelle des HTTP Protokolls zu nutzen ist das Vorschalten eines Apache Servers. Alternativ dazu kann auch der Zope Server dazu gebracht werden mittels HTTPS zu kommunizieren, aber die Konfiguration ist etwas komplizierter und wird im nachfolgenden Abschnitt behandelt.

In diesem Abschnitt wird nicht tief auf die Konfigurationsmöglichkeiten von Apache eingegangen. Es wird vielmehr beschrieben, wie das gesetzte Ziel, die Kommunikation über HTTPS, auf einfache Weise erreicht werden kann.

Als erste muss natürlich das Apache System sowie die `mod_ssl` Erweiterung installiert werden. Das genaue Vorgehen ist dabei Betriebssystem- sowie Distributionsabhängig. In einer RHEL oder CentOS Umgebung ist dazu lediglich ein

```
# yum install httpd mod_ssl
```

beziehungsweise in einer Ubuntu Umgebung ein

```
# apt install apache2
# a2enmod ssl
```

notwendig.

Jetzt muss die ssl Engine von Apache konfiguriert werden. Häufig erledigen die Paketmanager schon einiges an Arbeit und erzeugen ein self-signed Certificate. Dieses kann übernommen, oder mit einem offiziellen Zertifikat ersetzt werden.

Weiterhin muss in der ssl Konfiguration für den entsprechenden Virtual Host eine Proxy-Regel aufgenommen werden. Diese sieht in Prinzip folgendermaßen aus:

```
#
# Redirect to Zope
#
<IfModule mod_proxy.c>
    ProxyRequests Off
    ProxyPreserveHost On
    ProxyPass / http://127.0.0.1:8080/VirtualHostBase/\
        https/myvirtualhost:443/VirtualHostRoot/
    ProxyPassReverse / http://127.0.0.1:8080/VirtualHostBase/\
        http/https/myvirtualhost:443/VirtualHostRoot/
</IfModule>

<IfModule mod_headers.c>
    RequestHeader set X-Forwarded-Proto "https"
</IfModule>
```

Bitte beachten: Aus Gründe der Darstellung wurde die Zeile umgebrochen. Dies wird hier, wie so üblich, mit einem Backslash gekennzeichnet. In der tatsächliche Konfiguration sollten die beide Zeilen wieder, ohne Backslash und Whitespace, aneinandergehängt werden. Da der Zope Server auf denselben Host läuft als der vorgeschaltete Apache Server, wird die zu bearbeitenden Anfrage an die 127.0.0.1:8080 weitergeleitet. Als der eigene Hostname wurde myvirtualhost eingesetzt.

Im Header der Anfrage wird das Kommunikationsprotokoll auf https gesetzt.

Jetzt ist es nur noch wichtig jede Versuch mittles HTTP auf den Server zuzugreifen nach HTTPS umzulenken. Dies passiert in der globalen Konfiguration. Es wird in der Konfiguration des entsprechenden Virtual Hosts

```
Redirect permanent / https://myvirtualhost/
```

aufgenommen.

SSO für schedulix mit Zope

Einleitung

In kleinen Umgebungen ist die in schedulix integrierte Benutzerverwaltung sinnvoll und erlaubt auf einfacher Weise eine Trennung von Systemen. Dies ändert sich jedoch drastisch, wenn die Umgebungen größer werden. Um die Vielzahl der unterschiedlichen Systeme, deren Benutzer und einhergehenden Rechten im Überblick behalten zu können, ist eine zentrale Verwaltung unabdingbar.

In vielen Fällen wird eine solche Verwaltung mit Hilfe von Microsoft's Active Directory implementiert. Eine interessante Funktionalität ist dabei das Single Sign-On (SSO) Prinzip. Dabei wird eine Authentifizierung eines Benutzers bei der Anmeldung an seinem Arbeitsrechner durchgeführt. Bei einem Zugriff auf ein System welches SSO unterstützt, wird

keine erneute Password-Abfrage durchgeführt. Vielmehr wird mit Hilfe eines Tokens festgestellt, dass die Authentifizierung bereits erfolgt ist.

Abgesehen davon, dass dies eine Erleichterung für den Benutzer darstellt, werden in dem Prozess auch keine sensiblen Daten ausgetauscht, was zu einer deutlichen Erhöhung der Sicherheit führt.

Anhand einer Beispielumgebung wird in diesem Abschnitt gezeigt, wie eine Anbindung von schedulix an das Active Directory gemacht werden kann. Es wird einfach sein, anhand dieser Beschreibung die Abbildung in die eigene Umgebung zu machen.

In der Beispielumgebung ist der schedulix Server sowie der Zope Server auf ein CentOS 7 Linux System installiert. Der Rechner heißt `centos7sso`. Im Netzwerk befindet sich auch ein Active Directory Server mit Namen `adserver`, sowie ein Windows Client. Beide Windows Maschinen befinden sich in der Windows Domäne `INDEPENDIT.DE`.

Vorgehen

Zuerst wird ein Apache Webserver installiert, mit den benötigten Modulen versehen, und konfiguriert. Ziel ist dabei nicht nur die SSO Funktionalität, sondern auch die Kommunikation über https. Der Apache Server wird über eine `proxy_html` Schnittstelle mit dem Zope Server kommunizieren. Da letztendlich auch der Scheduling Server eine Authentifizierung vornimmt, muss auch dieser von der SSO Situation in Kenntnis gesetzt werden.

Kerberos Installation und Konfiguration

Das SSO Protokoll baut intern auf Kerberos auf. Daher muss die dazu benötigte Software installiert und konfiguriert werden.

Die Installation ist am Einfachsten. In einer RHEL oder CentOS Umgebung reicht ein einfaches

```
yum install krb5-libs
yum install krb5-workstation
yum install sssd-krb5
yum install sssd-krb5-common
```

aus.

Die Datei `/etc/krb5.conf` enthält die Konfiguration der Kerberos Installation. In unserer Beispielumgebung sieht sie folgendermaßen aus:

```
includedir /etc/krb5.conf.d/

[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log

[libdefaults]
dns_lookup_realm = false
default_keytab_name = /etc/httpd/krb5.keytab
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
rdns = false
pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
default_realm = INDEPENDIT.DE
default_ccache_name = KEYRING:persistent:%{uid}

[realms]
INDEPENDIT.DE = {
    kdc = ADSERVER.INDEPENDIT.DE
    master_kdc = ADSERVER.INDEPENDIT.DE
    admin_server = ADSERVER.INDEPENDIT.DE
    default_domain = INDEPENDIT.DE
```

```

}

[login]
krb4_convert = true
krb4_get_tickets = false

[domain_realm]
.independit.de = INDEPENDIT.DE
independit.de = INDEPENDIT.DE

```

Die genannte Datei /etc/httpd/krb5.keytab kann auf dem Active Directory Server erzeugt und nach dem rüberkopieren dort gelöscht werden:

```

ktpass -princ HOST/centos7sso.independit.de@INDEPENDIT.DE -mapuser \
bicsuite@INDEPENDIT.DE -crypto RC4-HMAC-NT \
-ptype KRB5_NT_PRINCIPAL -pass "VerySecret" \
-out c:\temp\krb5.keytab
ktpass -princ HTTP/centos7sso.independit.de@INDEPENDIT.DE -mapuser \
bicsuite@INDEPENDIT.DE -crypto RC4-HMAC-NT \
-ptype KRB5_NT_PRINCIPAL -pass "VerySecret" \
-out c:\temp\krb5.keytab -in c:\temp\krb5.keytab

```

Es kann sein, dass anstelle von RC4-HMAC-NT eine andere Verschlüsselung, wie zum Beispiel AES256-SHA1 gewählt werden muss. Aus Sicherheitsgründen ist die Benutzung moderner Verschlüsselungsverfahren empfehlenswert.

Um später die Authorisierung durchführen zu können, benötigt der Apache Server einen Zugang zum Active Directory Server. Im obigen Beispiel wurde dazu den unprivilegierten Account bicsuite@INDEPENDIT.DE genutzt. Der Name an sich ist unwichtig. Als Passwort wurde VerySecret eingestellt. Ein Passwort, was leicht erraten werden kann, auch wenn's "SehrGeheim" ist. Selbstverständlich sollte in einer realen Umgebung ein sicheres Passwort benutzt werden.

Ob die Kerberos Konfiguration so weit richtig ist, kann mittels

```

KRB5_TRACE=/dev/stdout kinit -k -t krb5.keytab \
-p HTTP/centos7sso.independit.de

```

überprüft werden:

```

[root@centos7sso httpd]# KRB5_TRACE=/dev/stdout kinit -k \
-t krb5.keytab -p HTTP/centos7sso.independit.de
...: Getting initial credentials for \
HTTP/centos7sso.independit.de@INDEPENDIT.DE
...: Looked up etypes in keytab: aes256-cts
...: Sending unauthenticated request
...: Sending request (225 bytes) to INDEPENDIT.DE
...: Resolving hostname ADSERVER.INDEPENDIT.DE
...: Sending initial UDP request to dgram 192.168.123.45:88
...: Received answer (204 bytes) from dgram 192.168.123.45:88
...: Response was from master KDC
...: Received error from KDC: -1765328359/Additional \
pre-authentication required
...: Preauthenticating using KDC method data
...: Processing preauth types: PA-PK-AS-REQ (16), \
PA-PK-AS-REP_OLD (15), PA-ETYPE-INFO2 (19), \
PA-ENC-TIMESTAMP (2)
...: Selected etype info: etype aes256-cts, salt \
"INDEPENDIT.DEHTTPcentos8sso.independit.de", params ""
...: Retrieving HTTP/centos8sso.independit.de@INDEPENDIT.DE \
from FILE:krb5.keytab (vno 0, enctype aes256-cts) with \
result: 0/Success
...: AS key obtained for encrypted timestamp: aes256-cts/8EB8
...: Encrypted timestamp (for 1619424732.825845): plain ...
...: Preauth module encrypted_timestamp (2) (real) returned: \
0/Success

```

```

.... Produced preauth for next request: PA-ENC-TIMESTAMP (2)
.... Sending request (305 bytes) to INDEPENDIT.DE
.... Resolving hostname ADSERVER.INDEPENDIT.DE
.... Sending initial UDP request to dgram 192.168.123.45:88
.... Received answer (98 bytes) from dgram 192.168.123.45:88
.... Response was from master KDC
.... Received error from KDC: -1765328332/Response too big for \
    UDP, retry with TCP
.... Request or response is too big for UDP; retrying with TCP
.... Sending request (305 bytes) to INDEPENDIT.DE (tcp only)
.... Resolving hostname ADSERVER.INDEPENDIT.DE
.... Initiating TCP connection to stream 192.168.123.45:88
.... Sending TCP request to stream 192.168.25.3:88
.... Received answer (1706 bytes) from stream 192.168.123.45:88
.... Terminating TCP connection to stream 192.168.123.45:88
.... Response was from master KDC
.... Processing preauth types: PA-ETYPE-INFO2 (19)
.... Selected etype info: etype aes256-cts, salt \
    "INDEPENDIT.DEHTTPcentos8sso.independit.de", params ""
.... Produced preauth for next request: (empty)
.... AS key determined by preauth: aes256-cts/8EB8
.... Decrypted AS reply; session key is: aes256-cts/9218
.... FAST negotiation: unavailable
.... Initializing KCM:0:99729 with default princ \
    HTTP/centos8sso.independit.de@INDEPENDIT.DE
.... Storing HTTP/centos8sso.independit.de@INDEPENDIT.DE -> \
    krbtgt/INDEPENDIT.DE@INDEPENDIT.DE in KCM:0:99729
.... Storing config in KCM:0:99729 for \
    krbtgt/INDEPENDIT.DE@INDEPENDIT.DE: pa_type: 2
.... Storing HTTP/centos8sso.independit.de@INDEPENDIT.DE -> \
    krb5_ccache_conf_data/pa_type/krbtgt\INDEPENDIT.DE\
    @INDEPENDIT.DE@X-CACHECONF: in KCM:0:99729

```

(Aus Darstellungsgründen wurde der Output etwas gekürzt und umformatiert).

Apache Webserver und Module

Installation

Als nächste können jetzt Apache (httpd) sowie einige benötigte Module installiert werden. Unter RHEL/CentOS 7 kann entweder das Modul `mod_auth_kerb` oder das Modul `mod_auth_gssapi` benutzt werden, ab RHEL/CentOS 8 gibt es allerdings kein Support mehr für das Kerberos Modul, so dass hier zwingend die gssapi Schnittstelle benutzt werden muss.

Daher entweder

```

yum install httpd
yum install mod_ssl
yum install mod_ldap
yum install mod_proxy_html
yum install mod_auth_kerb

```

für das Kerberos Modul, oder

```

yum install httpd
yum install mod_ssl
yum install mod_ldap
yum install mod_proxy_html
yum install mod_auth_gssapi

```

für das gssapi Modul.

Konfiguration

Die Konfiguration von Apache liegt bei RedHat basierenden Systemen unter `/etc/httpd` sowie in einigen Unterverzeichnissen. Das kann bei anderen Distributionen anders sein, jedoch bleibt das Prinzip gleich.

Zuerst wird dafür gesorgt, dass ausschließlich über https mit dem Apache Webserver kommuniziert wird. Dazu müssen in der Datei `/etc/httpd/conf.d/ssl.conf` einige Einträge vorgenommen werden. In der vorliegenden Beispielumgebung sind dies:

```
ServerName centos7sso.independit.de:443
SSLCertificateFile /etc/pki/tls/certs/centos7sso.crt
SSLCertificateKeyFile /etc/pki/tls/private/centos7sso.key
```

Je nach Umgebung kann es natürlich sein, dass auch die Intermediate Certificates noch eine Rolle spielen.

In der Datei `etc/httpd/conf/httpd.conf` wird dann noch dafür gesorgt, dass eventuelle Anfragen auf dem Standard http Port auf den https Port umgeleitet werden:

```
<VirtualHost _default_:80>
    ServerName centos7sso.independit.de:80
    #
    # force the use of https
    #
    Redirect permanent / https://centos7sso.independit.de/
</VirtualHost>
```

Selbstverständlich sollte auch die Firewall informiert werden, zum Beispiel:

```
firewall-cmd --zone=public --add-service=https
firewall-cmd --zone=public --permanent --add-service=https
```

Die Authentifizierung des Benutzers sieht etwas komplizierter aus und muss mit Information aus der Umgebung angereichert werden. Dabei gibt es, abhängig von dem benutzten Modul, zwei ähnliche, dennoch unterschiedliche Konfigurationen.

Im Falle von `mod_auth_kerb` funktioniert folgende Konfiguration in der Beispielumgebung:

```
# Wenn AD User vielen Gruppen zugehörigen sind kann die
# FieldSize schnell ausgereizt werden und man rennt in einen Fehler
# Deshalb wird diese hier vergrößert
LimitRequestFieldSize 32768
<Location "/bicsuite">
    AuthType Kerberos
    KrbAuthRealms INDEPENDIT.DE
    KrbServiceName HTTP
    Krb5Keytab /etc/httpd/krb5.keytab
    KrbMethodNegotiate On
    KrbMethodK5Passwd Off
    require valid-user

    RewriteEngine on
    RewriteCond %{REMOTE_USER} (.*?)
    RewriteRule .* - [E=X_REMOTE_USER:%1]
    RequestHeader set REMOTE_USER %{X_REMOTE_USER}e
    RequestHeader set X-Remote-User %{REMOTE_USER}s

    SetHandler "proxy:http://127.0.0.1:8080"
    SetEnvIfNoCase ^Authorization$ "(.+)" HTTP_AUTHORIZATION=$1
</Location>
```

Die Konfiguration für `mod_auth_gssapi` sieht ähnlich, sogar einfacher aus:

```
<Location "/bicsuite">
    AuthType GSSAPI
    AuthName "INDEPENDIT.DE"
    GssapiCredStore keytab:/etc/httpd/krb5.keytab
    GssapiSSLOnly On
    GssapiLocalname Off

    Require valid-user
```

```

RewriteEngine on
RewriteCond %{REMOTE_USER} (.* )
RewriteRule .* - [E=X_REMOTE_USER:%1]
RequestHeader set REMOTE_USER %{X_REMOTE_USER}e
RequestHeader set X-Remote-User %{REMOTE_USER}s

SetHandler "proxy:http://127.0.0.1:8080/"
SetEnvIfNoCase ^Authorization$ "(.*)" HTTP_AUTHORIZATION=$1
</Location>

```

Wenn der Webbenutzer die Location bicsuite, oder natürlich eine Ressource unterhalb des bicsuite Folders anfordert, dann wird mittels Kerberos geprüft, ob er dazu berechtigt ist. Dazu wird die Domäne, in unserem Fall INDEPENDIT.DE benötigt, sowie auch eine Datei krb5.keytab, die im Konfigurationsverzeichnis von Apache abgelegt wurde. Wenn die Authentifizierung erfolgreich ist, wird die Anfrage mittels http an den Zope Server weitergeleitet. Eventuelle Authorization Headers werden dabei ebenfalls mitgegeben.

selinux

Wenn selinux aktiviert ist, ist es dem Apache Server untersagt selbst Socket-Verbindungen zu öffnen. Da dies aber für die Kommunikation zwischen Apache und Zope benötigt wird, muss dies erlaubt werden:

```
/usr/sbin/setsebool -P httpd_can_network_connect 1
```

Ergebnistest

Ob alles so weit gut gegangen ist, kann einfach getestet werden. Dazu wird im DocumentRoot, in diesem Fall /var/www/html, ein Verzeichnis, z.B. ssotest, angelegt. In dem Verzeichnis wird eine Datei index.html angelegt, mit z.B. folgendem Inhalt:

```

<DOCTYPE html>
<html>
  <body>

    <h1>My First Heading</h1>

    <p>Hello World</p>

  </body>
</html>

```

Für den Test muss jetzt noch vorübergehend die Apache Konfiguration angepasst werden. Dazu wird der location Abschnitt wie folgt modifiziert:

```

<Location "/ssotest">
  AuthType          Kerberos
  KrbAuthRealms     INDEPENDIT.DE
  KrbServiceName    HTTP
  Krb5Keytab        /etc/httpd/krb5.keytab
  KrbMethodNegotiate On
  KrbMethodK5Passwd Off
  require valid-user

  # SetHandler "proxy:http://127.0.0.1:8080"
  # SetEnvIfNoCase ^Authorization$ "(.*)" HTTP_AUTHORIZATION=$1
</Location>

```

Statt Location bicsuite wird das neu erzeugte Verzeichnis ssotest angesprochen. Weiterhin wird verhindert, dass Apache die Anfrage an Zope weiterleitet.

Wenn nach einem Neustart des Apache Servers von einer Windows Workstation auf die Seite <https://centos7sso/ssotest> zugegriffen wird, sollte das "Hello World" angezeigt werden. Der Versuch die Seite vom Server selbst aus abzuholen, wie zum Beispiel

mittels `wget https://localhost/ssotest`, sollte ein "401 Unauthorized" zur Folge haben.

So fern der Test erfolgreich war, sollten die Änderungen jetzt wieder zurückgenommen werden.

Zope Erweiterung und Konfiguration

Der zweite Mitspieler ist der Zope Server, der natürlich auch von seinem Glück erfahren soll. Auch Zope muss in der Lage sein, mit dem Active Directory Server zu reden. Dazu benötigt er das `python_ldap` Paket, dessen Installation wiederum das `openldap-devel` Paket benötigt:

```
yum install openldap-devel
cd $BICSUITEHOME/./software/Zope
bin/pip install python-ldap
```

Es empfiehlt sich die Vorbereitung von Zope auf die Verwendung von SSO vor dem Einspielen vom `SDMS.zexp` zu machen. Allerdings ist das Aufsetzen von SSO auch nachträglich möglich, bedarf dann aber ein paar extra Schritten. Insbesondere ist zu beachten, dass die URL, mit der die GUI angesprochen wird, sich im Vergleich mit einer normalen Installation ändert.

Product Installation

Damit Zope Anfragen im richtigen Kontext ausführen kann, wird eine Benutzererkennung benötigt. Ohne SSO erfolgt eine Anmeldung innerhalb von Zope und damit ist die Information vorhanden. Im SSO Fall bekommt Zope die Benutzerinformation von dem Apache Server übermittelt. Damit Zope mit der Information umgehen kann, wird eine Erweiterung benötigt.

Das Installieren geht einfach:

```
cd $BICSUITEHOME/./bicsuiteweb/Products
ln -s $BICSUITEHOME/bicsuite/zope/RemoteUserFolder .
```

Es ist wichtig den Port 8080 zwar für Apache, aber nicht von außen zugänglich zu machen. Mit Hilfe von `firewalld` kann das wie folgt aussehen:

```
firewall-cmd --permanent --zone=public --add-rich-rule='
rule family="ipv4"
source address="127.0.0.1/32"
port protocol="tcp" port="8080" accept'
```

Installieren von SDMS.zexp

Die Installation der GUI Anwendung läuft analog zu der Installation ohne SSO. Wichtig jedoch ist, dass die Installation diesmal nicht im Root-Folder sondern in einem Unterfolder erfolgt. Wie der Folder heißt, ist weniger wichtig, so fern die Namensgebung überall korrekt beibehalten wird.

Aus technischen Gründen werden zwei leere Folder, sowie ein Folder für die Installation der Software im Root-Folder von Zope benötigt. Dazu wird die Management Oberfläche von Zope, unter Umgehung des Apache Servers aufgerufen. In der Beispielumgebung wird dazu also die URL

`http://centos7sso.independit.de:8080/manage`

in den Browser angegeben. Wenn eine Login Aufforderung erscheint, müssen jetzt die Daten des Benutzers, der bei der Installation von Zope angegeben wurde, benutzt werden. Typischerweise ist das der Benutzer `sdmsadm`.

Zuerst wird jetzt ein Folder `bicsuite`, ein Folder `web`, sowie ein Folder `GUI` angelegt. Und es wird ein Property `SDMSROOT` auf den Wert `/bicsuite/GUI` erzeugt. Später wird die GUI dann unter der URL

<https://centos7sso.independit.de/bicsuite/GUI/SDMS>

aufgerufen werden können.

Jetzt folgen alle Schritte wie bei der normalen Installation, allerdings wird SDMS.zexp in den Folder GUI importiert. Auch das Anlegen der Folders Custom und User erfolgt im Folder GUI.

Im Custom Folder gibt es ein Konfigurationsscript namens SDMSServers, in dem festgehalten wird, welche Scheduling Servers von der Zope Instanz aus bedient werden können. Zope muss wissen, dass SSO benutzt werden soll:

```
#
# define all accessible SDMS Servers here
#
return {
  'DEFAULT' : {
    'HOST'    : 'localhost',
    'PORT'    : '2506',
    'VERSION' : 'BASIC',
    'CACHE'   : 'Y',
    'TIMEOUT' : '60',
    'SSO'     : True, # <-- Benutze SSO
    'SSL'     : 'N',
    #
    # Edit following Options if SLL set to 'Y'
    #
    # 'TRUSTSTORE' : 'truststore.pem',
    # 'KEYSTORE'  : 'keystore.pem'
  }
}
```

Zum Schluss wird noch ein Objekt vom Typ RemoteUserFolder im Folder GUI angelegt.

Konfiguration der SDMS Anwendung

Unter \$BICSUITEHOME/etc liegt eine Datei namens ZopeSSO.conf.template. Diese wird nach dem Konfigurationsverzeichnis \$BICSUITECONFIG kopiert und dabei nach ZopeSSO.conf umbenannt.

Die Datei ist ziemlich selbsterklärend. Es geht los mit allgemeinen Einstellungen, die dann pro Domäne, oder Server, überschrieben werden können.

In der Beispielumgebung wurde mit folgenden Einstellungen gearbeitet:

```
# =====
# ZopeSSO.conf.template
#
# Copy this file to the BICSUITECONFIG directory and edit it according
# to your needs
# At least all properties set to <TO_BE_CONFIGURED> have to be set.
# =====
# Configurations for handling SSO for the BICsuite web frontend
#
# WARNING:
# This file contains credentials for LDAP and BICsuite ADMIN access.
# Make this file only readable for the user running the Zope
# application server
{
  #=====
  # General configurations which can be (partially) overridden by
  # domain-specific or server-specific configurations
  #-----
  # Defaults for domain specific settings if not set in the DOMAINS
  # section
  #-----
  # WebNameCase defines how names for Zope authenticated user names
  # are converted
  # 'UPPER' convert to upper case
```

```

# 'LOWER' convert to lower case
# 'MIXED' no conversion (default)
'WebNameCase' : 'MIXED',
#-----
# WebAutoCreateUsers indicates if AD users should be created
# automatically as BICsuite frontend users. If WebUseLdapGroups,
# is set to True, only AD users who are members of the UserGroup
# and/or ManagerGroup below will be allowed.
# True
# False (default)
'WebAutoCreateUsers' : True,
#-----
# WebUseLdapGroups indicates if ldap groups should be used to
# detect whether an AD user is allowed to log in to the BICsuite
# web frontend
# True
# False (default)
'WebUseLdapGroups' : True,
#-----
# WebIncludeDomainNames indicates if Domain Names should be part
# of web user identifiers
# True or False
# defaults to False
'WebIncludeDomainNames' : False,
#-----
# WebUserGroup allowed to log in in via SSO
# defaults to 'BICSUITE_WEB_USER'
'WebUserGroup' : 'bicsuite',
#-----
# manager group granting manage privilege on Zope website
# defaults to 'BICSUITE_WEB_MANAGER'
'WebManagerGroup' : 'bicsuite_admin',
#-----
# WebGroupCheckIntervall is the time in minutes after which ldap
# group assignments for a BICsuite web server are checked again
# defaults to 60 (1 hour)
'WebGroupCheckIntervall' : 60
#-----
# Defaults for server-specific settings if not set in the SERVERS
# section
#-----
# ServerIncludeUserDomainNames indicates if domain names should be
# part of user identifiers
# True or False
# defaults to False
'ServerIncludeUserDomainNames' : False,
#-----
# ServerIncludeGroupDomainNames indicates if domain names should
# part of group identifiers
# True or False
# defaults to False
'ServerIncludeGroupDomainNames' : False,
#-----
# ServerUserNameCase defines how names for BICsuite are converted
# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
'ServerUserNameCase' : 'UPPER',
#-----
# ServerGroupNameCase defines how names for BICsuite groups are
# converted
# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
'ServerGroupNameCase' : 'UPPER',
#-----

```

```

# ServerAutoCreateUsers indicates if AD users should be created
# automatically. If ServerUseLdapGroups is True, only AD users
# who are a member of any AD group named
# <ServerBicsuitePrefix>_<ServerName>_<groupname> are allowed
# True or False
# defaults to False
'ServerAutoCreateUsers' : True,
#-----
# ServerUseLdapGroups indicates if AD groups should be used
# True or False
# defaults to False
'ServerUseLdapGroups' : True,
#-----
# ServerBicsuitePrefix is the prefix used for AD groups. Groups
# called other than <ServerBicsuitePrefix>_<ServerName>_<groupname>
# are ignored
# defaults to 'BICSUITE'
'ServerBicsuitePrefix' : 'bicsuite',
#-----
# ServerName is the server name used for AD groups. Groups called
# other than <ServerBicsuitePrefix>_<ServerName>_<groupname>
# are ignored
# defaults to 'DEFAULT'
'ServerName' : 'centos7sso',
#-----
# ServerDefaultGroupSuffix
# Suffix used to decide whether a AD group should be the default
# group defaults to '_ISDEFAULT'
'ServerDefaultGroupSuffix' : '_ISDEFAULT',
#=====
# Domain-specific configurations independent of the BICSuite
# server. Accessing users from domains not configured here will not
# be able to log on to the BICSuite web frontend via SSO
#-----
'DOMAINS' : {
    # domain name as in <DOMAIN_NAME>\UserName
    'INDEPENDIT.DE' : {
        #-----
        # Domain-specific settings or one BICSuite server
        #-----
        # ldap server and base to get group membership from
        # Example:
        # 'LdapServer' : 'ldap://192.168.0.1',
        'LdapServer' : 'ldap://adserver.independit.de',
        # Example:
        # 'LdapBaseDn' : 'DC=INDEPENDIT,DC=dieter,DC=de',
        'LdapBaseDn' : 'DC=INDEPENDIT,DC=de',
        #-----
        # ldap credentials to use for group membership retrieval
        # Example
        # 'LdapUsername' : 'Administrator@INDEPENDIT.DIETER.DE',
        'LdapUsername' : 'bicsuite_admin',
        'LdapPassword' : 'G0H0ME-123',
        #-----
        # WebNameCase defines if names for Zoep authenticated user
        # names have to be converted to
        # 'UPPER' convert to upper case (default)
        # 'LOWER' convert to lower case
        # 'MIXED' no conversion
        'WebNameCase' : 'UPPER',
        #-----
        # WebAutoCreateUsers indicates if AD users should be created
        # automatically as BICSuite frontend users. If
        # UseLdapWebGroups is set to True, only AD users who are
        # members of the UserGroup and/or ManagerGroup below will
        # be allowed.
        # True
        # False (default)

```

```

'WebAutoCreateUsers' : True,
#-----
# WebUseLdapGroups indicates if Ldap groups should be used
# to detect whether AD user is allowed to log in to the
# BICsuite web frontend
# True
# False (default)
# 'WebUseWebGroups' : False,
#-----
# WebIncludeDomainNames indicates if Domain Names should be
# part of web user identifiers
# True or False
# defaults to False
'WebIncludeDomainNames' : False,
#-----
# WebUserGroup allowed to log in in via SSO
# defaults to 'BICSUITE_WEB_USER'
'WebUserGroup' : 'bicsuite',
#-----
# manager group granting manage privilege on Zope website
# defaults to 'BICSUITE_WEB_MANAGER'
'WebManagerGroup' : 'bicsuite_admin',
#-----
# WebGroupCheckIntervall is the time in minutes after which ldap
# group assignments for a BICsuite web server are
# checked again
# defaults to 60 (1 hour)
'WebGroupCheckIntervall' : 60
#-----
}
},
#=====
# BICsuite server-specific configurations
#-----
'SERVERS' : {
#-----
# For every BICsuite server to be accessed via SSO, the following
# section must be created with hostname:port
# Example: localhost:2506
'localhost:2506' : {
#-----
# General configuration for a BICsuite server independent
# of the login domain
#-----
# login credentials for a BICsuite admin user who is allowed
# to manage users and group. Used also to connect to BICsuite
# before sending the 'alter session set user' command when
# executing statements vis-a-vis BICsuite for a user
'AdminUser'      : 'SYSTEM',
'AdminPassword'  : 'GOHOME',
#-----
# Defaults for server-specific settings if not set in the
# DOMAINS section
#-----
# ServerIncludeUserDomainNames indicates if domain names
# should be part of user identifiers
# True or False
# defaults to False
'ServerIncludeUserDomainNames' : False,
#-----
# ServerIncludeGroupDomainNames indicates if domain names
# should be part of group identifiers
# True or False
# defaults to False
'ServerIncludeGroupDomainNames' : False,
#-----
# ServerUserNameCase defines how names for BICsuite users
# are converted

```

```

# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
'ServerUserNameCase' : 'UPPER',
#-----
# ServerGroupNameCase defines how names for BICsuite groups
# are converted
# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
'ServerGroupNameCase' : 'UPPER',
#-----
# ServerAutoCreateUsers indicates if AD users should be
# created automatically. If ServerUseLdapGroups is True
# only AD users who are a member of any AD group named
# <ServerBicsuitePrefix>_<ServerName>_<groupname> are allowed
# True or False
# defaults to False
'ServerAutoCreateUsers' : True,
#-----
# ServerUseLdapGroups indicates if AD groups should be used
# True or False
# defaults to False
'ServerUseLdapGroups' : True,
#-----
# ServerBicsuitePrefix is the prefix used for AD groups.
# Groups called other than
# <ServerBicsuitePrefix>_<ServerName>_<groupname> are ignored
# defaults to 'BICSUITE'
'ServerBicsuitePrefix' : 'BICSUITE',
#-----
# ServerName is the server name used for AD groups. Groups
# called other than
# <ServerBicsuitePrefix>_<ServerName>_<groupname> are ignored
# defaults to 'DEFAULT'
'ServerName' : 'centos7sso',
#-----
# ServerDefaultGroupSuffix
# Suffix used to decide whether a AD group should be the
# default group
# defaults to '_ISDEFAULT'
'ServerDefaultGroupSuffix' : '_ISDEFAULT',
#-----
# Domain-specific configurations for this BICsuite server
#-----
'DOMAINS' : {
    # optional server and domain-specific configuration
    # overriding server and base defaults
    'INDEPENDIT.DE' : {
        #-----
        # Server-specific settings for this domain
        #-----
        # ServerIncludeUserDomainNames indicates if
        # domain names should be part of user
        # identifiers
        # True or False
        # defaults to False
        'ServerIncludeUserDomainNames' : False,
        #-----
        # ServerIncludeGroupDomainNames indicates if
        # domain names should be part of group
        # identifiers
        # True or False
        # defaults to False
        'ServerIncludeGroupDomainNames' : False,
        #-----
    }
}

```



```

# ServerUserNameCase defines how names for
# BICsuite users
# are converted
# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
# 'ServerUserNameCase' : 'UPPER',
#-----
# ServerGroupNameCase defines how names for
# BICsuite groups are converted
# UPPER case
# LOWER case
# MIXED case (don't convert them)
# defaults to 'UPPER'
# 'ServerUserNameCase' : 'UPPER',
#-----
# ServerAutoCreateUsers indicates if AD users
# should be created automatically.
# If ServerUseLdapGroups is True, only AD users
# who are a member of any AD group named
# <ServerBicsuitePrefix>_<ServerName>_<groupname>
# are allowed
# True or False
# defaults to False
# 'ServerAutoCreateUsers' : False,
#-----
# ServerUseLdapGroups indicates if AD groups
# should be used
# True or False
# defaults to False
# 'ServerUseLdapGroups' : False,
#-----
# ServerBicsuitePrefix is the prefix used for AD
# groups. Groups called other than
# <ServerBicsuitePrefix>_<ServerName>_<groupname>
# are ignored
# defaults to 'BICSUITE'
# 'ServerBicsuitePrefix' : 'BICSUITE',
#-----
# ServerName is the server name used for AD
# groups. Groups called other than
# <ServerBicsuitePrefix>_<ServerName>_<groupname>
# are ignored
# defaults to 'DEFAULT'
# 'ServerName' : 'DEFAULT',
#-----
# ServerDefaultGroupSuffix
# Suffix used to decide whether a AD group
# should be the default group
# defaults to '_ISDEFAULT'
# 'ServerDefaultGroupSuffix' : '_ISDEFAULT',
#-----
    }
}
}
}
}

```

Konfiguration des schedulix Servers

Auch der schedulix Server braucht noch etwas Konfiguration, damit bekannt ist, wie er sich im Falle von SSO Anmeldungen verhalten soll. Wichtig dabei ist, dass die Einstellungen zu den Einstellungen des Zope Servers passen. In der Beispielumgebung wurden die relevanten Parameter wie folgt gesetzt:

```
#
# SSOincludeDomainNames indicates if Domain Names should
# be part of Group/User Identifiers
# default = false
#
SSOincludeDomainNames=false

#
# SSOautoCreateUsers indicates if AD users should be
# created automatically
# default = false
#
SSOautoCreateUsers=true

#
# SSOautoCreateGroups indicates if AD groups should be
# created automatically
# default = false
#
SSOautoCreateGroups=true

#
# SSOuseADGroups indicates if AD groups should be used
# or not
# default = false
#
SSOuseADGroups=true

#
# SSOserverName is the name of the server (within AD);
# this is used to filter out groups
#
SSOserverName=centos7sso

#
# SSObicsuitePrefix is the prefix used for AD groups.
# Groups called otherwise than
# <SSObicsuitePrefix>_<SSOserverName>_<groupname>
# are ignored
#
SSObicsuitePrefix=bicsuite

#
# SSOnameCase defines if names have to be converted to
# UPPER case (= default value; make identifier case
# insensitive if they adhere to BICsuite
# naming standards)
# LOWER case
# MIXED case (= don't convert them)
#
SSOnameCase=UPPER
```

Einstellungen an Benutzeroberfläche

Damit SSO funktioniert, muss der verwendete Browser wissen, dass er die Credentials dem Apache Server melden soll. Je nach Browser ist jedoch die Konfiguration etwas unterschiedlich. In den nächsten Abschnitten wird für einige häufig benutzte Browser erklärt welche Einstellungen gesetzt werden müssen. Die Reihenfolge ist rein alphabetisch.

Chrome und Microsoft Edge

Bei Chrome und Edge ist die Prozedur identisch. In Windows Systemsteuerung → Netzwerk und Internet → Internetoptionen → Tab Sicherheit wählen Sie das lokale Intranet. Nach einem Klick auf den Sites und anschließend den Erweitert Button, tragen Sie den Hostname des Apache Servers ein.

Firefox

In Firefox geben Sie die URL `about:config` ein. Dann suchen Sie nach dem Wort `negotiate`. Einer der Suchergebnisse sollte

```
network.negotiate-auth.trusted-uris
```

sein.

Ein Doppelklick erlaubt die Eingabe eines Wertes. Dabei wird als Wert der Hostname des Apache Servers eingetragen. In der Beispielumgebung ist das `centos7sso.independit.de`.

Nach dem Speichern und einem Neustart des Browsers sollte die GUI ohne Passwort-Eingabe erreichbar sein.

Administration des Zope Servers

Für die Administration des Zope Servers wird ein von der SSO Logik getrennter Zugang benötigt. Dies liegt darin begründet, dass die Benutzer die mittels SSO einen Zugang bekommen anders behandelt werden als "normale" Zope Benutzer.

Dennoch ist es wünschenswert, auch die Administration über eine gesicherte Verbindung abzuwickeln. Dies ist, mit einigen wenigen Einschränkungen, auch problemlos möglich. Prinzipiell wird, wie bereits beschrieben, der Apache Server als Reverse Proxy aufgesetzt. Allerdings dürfen Anfragen die `/bicsuite` referenzieren nicht übersetzt werden, da hier die SSO Logik greifen soll.

Damit ergibt sich folgende Ergänzung der Apache Konfiguration:

```
ProxyRequests Off
ProxyPreserveHost On
ProxyPass / http://127.0.0.1:8080/VirtualHostBase/https/\
centos7sso.independit.de:443/VirtualHostRoot/
ProxyPassReverse / http://127.0.0.1:8080/VirtualHostBase/\
http/https/centos7sso.independit.de:443/VirtualHostRoot/
RequestHeader set X-Forwarded-Proto "https"
```

Bitte beachten: Aus Gründen der Darstellung wurde die Zeile umgebrochen. Dies wird hier, wie so üblich, mit einem Backslash gekennzeichnet. In der tatsächlichen Konfiguration sollten die beiden Zeilen wieder, ohne Backslash und Whitespace, aneinandergehängt werden.

Diese Ergänzung kann direkt hinter den zuvor zugefügten Anweisungen in `ssl.conf` vorgenommen werden.

Es ist ratsam für administrative Aufgaben einen anderen als den Standard Browser zu benutzen, sonst entstehen leicht unerwünschte Nebeneffekte. Man kann nun mal nicht gleichzeitig als zwei verschiedene Benutzer in einem Browser an einer Zope Instanz angemeldet sein.