independIT Integrative Technologies GmbH
Bergstraße 6
D–86529 Schrobenhausen

schedulix

Installation Guide
Release 2.11

Dieter Stubler Ronald Jeninga

August 6, 2025

Copyright © 2025 independIT GmbH

Legal notice

This work is copyright protected

Copyright © 2025 independIT Integrative Technologies GmbH
All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronically or mechanically, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner.

Contents

1 Requirements

Compile environment

The following software is required to create the executables from the source package that are needed on a Linux system:

- Oracle(Sun) Java 1.8 JDK or later
  http://www.oracle.com/technetwork/java/index.html
  Alternatively including an OpenJDK 1.8 or later
  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
  The jay executable is provided in the package. The original sources and executables can be found here.
  http://www.cs.rit.edu/~ats/projects/lp/doc/jay/package-summary.html

  Important: the jay executable requires 32-bit libraries. They will have to be installed additionally on 64-bit systems.
- Eclipse SWT
  The package includes several examples for which SWT has to be installed. To make sure that the compile does not crash and the examples, function properly requires an Eclipse SWT.
  http://www.eclipse.org/swt
- Java Native Access (JNA)
  In order to avoid the use of a JNI library we use the JNA library from version 2.6 and later.
  https://github.com/twall/jna

In many cases, the required software packages can be easily installed using a Package Manager such as yum, rpm or dpkg.

schedulix Server

The following software is required to install the schedulix server:

- A for this architecture suitable schedulix-2.11.tgz
- Oracle(Sun) Java 1.8 SE JRE
  http://www.oracle.com/technetwork/java/index.html
  Alternatively an OpenJDK 1.8 or later
  http://openjdk.java.net
- One of the following RDBMS systems with the corresponding JDBC interface:
  - PostgreSQL

    http://www.postgresql.org

    JDBC for 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
  The package includes several examples for which SWT has to be installed. http://www.eclipse.org/swt If you don't want to install the examples, the SWT package is not required.

schedulix Client

The following software is required to install a schedulix client:

- A for this architecture suitable schedulix-2.11.tgz
- Oracle(Sun) Java 1.8 SE JRE
  http://www.oracle.com/technetwork/java/index.html
  Alternatively an OpenJDK 1.8 or later
  http://openjdk.java.net
- Eclipse SWT
  The package includes several examples for which SWT has to be installed. http://www.eclipse.org/swt If you don't want to execute the examples on this client, the SWT package is not required.

- Java Native Access (JNA)
  In order to avoid the use of a JNI library we use the JNA library from version 2.6 and later. This library is only required for the Jobserver.
  https://github.com/twall/jna

Zope Application Server

The web front end is provided by the Zope Application Server. The following software is required to install the Zope server:

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

2 Compiling the system

General preparation

Sensitive software should be installed under a separate account. This simplifies the administration and protects against abuse. In this guide it is assumed that the conversion and installation take place using the account `schedulix`. The home directory is assumed to be `/home/schedulix`. These are naturally just suggestions. It is not technically necessary to use them, although the guide will have to be interpreted accordingly if different parameters are being used.
How to create a user is described in the installation chapter on page 9.

Compile

To successfully translate the system after the required packages have been installed, some environment variables have to be set before "make" can do the actual work. Because no special privileges are required either for the conversion or the installation these are done under the user `schedulix`.

1. Download of the Schedulix source distribution from github
   All files needed for compilation and installing the system are available in github and can be retrieved using the following command:

   ```
   cd $HOME
   git clone https://github.com/schedulix/schedulix.git \
      -b v\versionnumber schedulix-\versionnumber
   ```

   Afterwards all the files of the schedulix source distribution will be stored in the subdirectory

   ```
   $HOME/schedulix-\versionnumber
   ```

2. Set the environment variables

   You now have to set some environment variables. The following commands are from an installation using a CentOS (`http://www.centos.org`) Linux distribution. In many cases the commands can be taken over as given here, but they are dependent upon the specific Linux distribution and the installed software.

```
export SDMSHOME=/home/schedulix/schedulix-\versionnumber
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
```

It is advisable to add these settings to `.bashrc` at least until the compilation of the system has been completed.

3. make
   All that remains to be done now is the actual compilation of the system.

```
cd ~/schedulix-\versionnumber/src
make
```

   In case of a repeated compilation attempt it is adviseable to enter `make new` instead of `make`.

   In the last operation a jar file is created and saved under `~/schedulix/lib`.

4. Create `~/schedulix-\versionnumber.tgz`

```
cd $HOME
tar czf schedulix-\versionnumber schedulix-\versionnumber
```

3 Installation in a Linux environment

Installing the schedulix Server

The installation of the schedulix Scheduling Server is uncomplicated and only requires a few simple actions which are explained in the following.
Where (example) commands are shown, the prompt is usually indicated with `$`. These commands are then executed under the `schedulix` account, which needs to be created. In some cases, the privileged `root` account is required. This is indicated by a `#` as prompt.

1. Create the user `schedulix`

   It is not necessary to name the user schedulix. This means that the name can also be modified for any convention. In this document it is assumed that the user is called schedulix.

   Under the Ubuntu distribution of Linux, a user can be created as follows:

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

   All the following operations are executed under the user schedulix except where a different user is explicitly stated.

2. Download and install a database management system supported by schedulix.

   schedulix for Linux currently supports the following systems:

   - Postgres (page 16)
   - MySQL (page 18)
   - Ingres (page 19)

   Reference is made to the appropriate sections regarding the installation of the chosen database system as well as how to modify the configuration of the schedulix Enterprise Scheduling System.

3. Unpack the software

   Unpack the tar archive in the schedulix home directory. For instance:

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

   Create a symbolic link:

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

4. Create the configuration

   a) User environment

   The following variables have to be set to be able to work with the schedulix system:

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

   It has proved to be good practice to save the system configuration in a different folder to the installation directory. This will make it substantially easier to upgrade the system later. Since the variables of all the system's users have to be set, it may be sensible to write the assignments (and exports) to a separate file and then to source this in `.profile` or `.bashrc`.

   b) Software environment

   Several templates for the configuration files that should be used as a basis for the system configuration can be found under `$BICSUITEHOME/etc`. These have to be copied without the ".template" extension to the directory `$BICSUITECONFIG`.

   For instance:

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

   Afterwards, the files obviously have to be modified to accommodate the environment.

   The file `bicsuite.conf` configures some default settings and does not usually have to be modified. However, it may be worth considering running the system logging in a different folder to the installation directory. In this case it is only necessary to change the variable `BICSUITELOGDIR` accordingly. The directory set in `BICSUITELOGDIR` must exist.

   The file `java.conf` describes the Java environment that is to be used. In particular, the path to the JDBC driver must be entered. The memory configuration of the server is also regulated here. Even in large-scale environments it is usually only necessary to adapt the variable `BICSUITEMEM`.

   The file `server.conf` contains the server configuration. The settings for connecting the schedulix Scheduling Server to its RDBMS repository

have to be modified. More details about this can be found in the respective chapter on the RDBMS that is being used.
In this file it is also necessary to change the `hostname` property to the server's hostname or IP address.

The file `jobserver.conf` is not required here, but serves as a template for the jobserver configuration.

5. Set up the database

   Follow the instructions for setting up the database system dependent upon which system you want to use.

   For

   - Ingres, see page 19,
   - MySQL, see page 18, and for
   - PostgreSQL, see page 16.

6. Start the server

   The installation is now more or less complete. You just have to start the server and load the examples as required.

   The server can be started with

   ```
   $ server-start
   ```

7. Create the file `.sdmshrc`

   The file `.sdmshrc`, if present, is read by all the schedulix command line tools for populating the command line parameters. In the following it is assumed that this file exists and that the correct values have been set for the users, password, host and port. The file `.sdmshrc` is created in the Linux user's home directory.

   Here is an example of the contents:

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

   Important: Since the file contains the data for accessing the Scheduling Server, the file rights should be set so that only its owner can read the file.

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

8. Install the convenience package

   The convenience package installs a commonly used configuration for an exit state model.

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

9. Install the examples (optionally)

   The installation routine for the examples comprises two parts. First of all, three so-called jobservers that are required for the subsequent workflow definitions are created. Sample workflow definitions are then loaded onto the server.

   a) Create the jobservers

      It just takes one script to create the jobservers:

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

   b) Load the workflow definitions

      The following commands are entered to load the workflow definitions:

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

   Because the examples assume that the jobservers have already been created, the above sequence is mandatory.

Installing a schedulix Client

The installation of a schedulix scheduling client is easy and requires only a few simple actions which are explained in the following.
Where (example) commands are shown, the prompt is usually indicated with `$`. These commands are then executed under the `schedulix` account, which needs to be created. In some cases, the privileged `root` account is required. This is indicated by a `#` as prompt.

1. Creating the user `schedulix`
   It is not necessary to name the user schedulix. This means that the name can also be modified for any convention. In this document it is assumed that the user is called schedulix.

   Under the Ubuntu distribution of Linux, a user can be created as follows:

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

   All the following operations are executed under the user schedulixexcept where a different user is explicitly stated.

2. Unpack the software

   Unpack the tar archive in the schedulix home directory. For instance:

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

   Create a symbolic link:

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

3. Create the configuration

   a) User environment

      The following variables have to be set to be able to work with the schedulix system:

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

      It has proved to be good practice to save the system configuration in a different folder to the installation directory. This will make it substantially easier to upgrade the system later. Since the variables of all the system's users have to be set, it may be sensible to write the assignments (and exports) to a separate file and then to source this in `.profile` or `.bashrc`.

   b) Software environment

      Several templates for the configuration files that should be used as a basis for the system configuration can be found under `$BICSUITEHOME/etc`.

      For a client installation we need the files `bicsuite.conf` and `java.conf`. These have to be copied without the ".template" extension to the directory `$BICSUITECONFIG`.

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

      The file `bicsuite.conf` configures some default settings and does not usually have to be modified.

      The file `java.conf` describes the Java environment that is to be used and usually does not require any changes.

4. Create the file `.sdmshrc`

   The file `.sdmshrc`, if present, is read by all the schedulix command line tools for populating the command line parameters. In the following it is assumed

that this file exists and that the correct values have been set for the users, password, host and port. The file `.sdmshrc` is created in the Linux user's home directory.

Here is an example of the contents:

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

Important: Since the file contains the data for accessing the Scheduling Server, the file rights should be set so that only its owner can read the file.

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

Jobserver Sample Installation

In the following a sample installation of a schedulix jobserver will be be shown.

Scenario

On the machine `machine_42` a jobserver shall execute processes as user `arthur`. The HOME directory of the user is `/home/arthur`.
The schedulix server is installed on the machine `scheduling_server` and is listening on port 2506. The system passwort is `G0H0ME`.

Requirements

On the machine `machine_42`, a schedulix client was installed in the HOME directory `/home/schedulix`.
The user `arthur` has the following access privileges on the client installation files:

- Read access on the files `java.conf` and `bicsuite.conf` in `$BICSUITECONFIG` and all files under `/home/schedulix/lib`
- Read and execute privileges on all files in `/home/schedulix/bin`

Installation

To enable a jobserver to connect to the schedulix scheduling server, the jobserver has to be configured within schedulix scheduling server. In the following we execute the neccesary steps using the `sdmsh` command line utility.
Alternatively this could also be done using the Web GUI.

1. Login as user `arthur` into the machine `machine_42`

2. Setting of the shell environment variables (`.bashrc`).

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

3. Testing the environment

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

   A `SDMS>` prompt should appear (use 'exit' to close sdmsh).

4. Create directories

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

5. Create a Scope for machine `machine_42` using sdmsh

```
SDMS> CREATE OR ALTER SCOPE GLOBAL.'MACHINE_42'
 WITH
  CONFIG = (
   'JOBEXECUTOR' = '/home/schedulix/schedulix/bin/jobserver',
   'HTTPHOST' = 'machine_42'
  );
```

   All directory paths must be full qualified. The use of shell environment variables is not supported.

6. Create a jobserver using sdmsh

```
SDMS> CREATE OR ALTER JOB SERVER GLOBAL.'MACHINE_42'.'ARTHUR'
 WITH
  PASSWORD =  'dent',
  NODE = 'machine_42',
  CONFIG = (
   'JOBFILEPREFIX' = '/home/arthur/taskfiles/',
   'DEFAULTWORKDIR' = '/home/arthur/work',
   'HTTPPORT' = '8905',
   'NAME_PATTERN_LOGFILES' = '/home/arthur/work/.*\\.log'
  );
```

The `HTTPPORT` is neccessary for the display of job logs and can be selected freely but must be unique for all jobserves on the same machine. All directory paths must be full qualified. The use of shell environment variables is not supported.

7. Create the Named Resource for the jobserver using sdmsh

```
SDMS> CREATE OR ALTER NAMED RESOURCE RESOURCE.'JOBSERVERS'
      WITH USAGE = CATEGORY;
SDMS> CREATE NAMED RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42'
      WITH USAGE = STATIC;
```

8. Create an Environment for the jobserver using sdmsh

```
SDMS> CREATE ENVIRONMENT 'ARTHUR@MACHINE_42'
      WITH RESOURCE = (RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42');
```

9. Create the Resource in the jobserver using sdmsh

```
SDMS> CREATE RESOURCE RESOURCE.'JOBSERVERS'.'ARTHUR@MACHINE_42'
      IN GLOBAL.'MACHINE_42'.'ARTHUR' WITH ONLINE;
```

10. Create the configuration file `$HOME/etc/jobserver.conf` for the jobserver containing the following:

```
RepoHost= scheduling_server
RepoPort= 2506
RepoUser= "GLOBAL.'MACHINE_42'.'ARTHUR'"
RepoPass= dent
```

11. Start the jobserver

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

    If the jobserver should start automatically at boot time, the system administrator has to configure this accordingly.

Installation with Postgres

Introduction

This guide does not claim to precisely describe the installation of the database system. Details about this can be found in the Postgres documentation. Normally, though, this guide should be adequate for performing a ”standard” installation.

Installation

1. Download and install the latest version of Postgres

   A Postgres package is normally provided for every Linux distribution. This package, as well as a package for the JDBC driver for Postgres, should usually be easy to install.

2. Configure the file `pg_hba.conf`

   To enable the schedulix Scheduling Server to identify itself to PostgreSQL with a user and password, the following line has to be added to the Postgres configuration file `pg_hba.conf`. This file is usually found under `/var/lib/pgsql/<version>/data`:

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

   PostgreSQL then has to be restarted.

3. Create the Postgres user `bicsuite`

   Log on as the Postgres user and run the command `createuser` as shown in the example (version 8):

   ```
   $ 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
   ```

   or, in the case of version 9:

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

   The entered password will be required again later.

4. Create the repository database `bicsuitedb`

   When you are logged on as the user schedulix, create the database for the repository as shown in the following example:

   ```
   $ createdb schedulixdb
   ```

5. Create and initialise the database tables

   To create the required database schema, switch to the schedulix sql directory and call the Postgres utility `psql` as shown in the following example:

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

6. Configure the database connection in the schedulix Server configuration file `$BICSUITECONFIG/server.conf`

   Change the following properties as shown here:

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

The `DbUrl` is to a certain degree dependent upon which version of PostgreSQL is installed. Under Version 8 this is

```
DbUrl=jdbs:postgresql:schedulixdb
```

7. Configure the schedulix Java Class Path for the Postgres JDBC

   Now all you have to do is to append the path to the Postgres JDBC to `CLASSPATH` in the configuration file `$BICSUITECONFIG/java.conf`.

   For instance:

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

Installation with MySQL

Introduction

This guide does not claim to precisely describe the installation of the database system. Details about this can be found in the MySQL documentation. Normally, though, this guide should be adequate for performing a "standard" installation.

Installation

1. Download and install the latest version of MySQL.

   Ready-to-use MySQL packages are available for most Linux distributions. These can be simply installed using the appropriate tools.

   During this installation, you will be prompted to enter a password for the MySQL root user (not to be confused with the Linux root user). This password is required again in the next step.

   Because schedulix sets up a JDBC connection to access the database, the MySQL JDBC driver has to be installed as well.

2. Create the MySQL user `bicsuite` and the database schedulixdb

   Start the Utility `mysql` utility and log on as the MySQL root user to create the user schedulixand the database schedulixdb:

   ```
   $ 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: 5.1.54-1ubuntu4 (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_password';
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. Create and initialise the database tables

   Run the following commands:

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

4. Configure the database connection in the `$BICSUITECONFIG/server.conf` configuration file

   Change the following properties as shown here:

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

5. Configure the schedulix Java Class Path for the MySQL JDBC

   Now all you have to do is to append the path to the MySQL JDBC to `CLASSPATH` in the configuration file `$BICSUITECONFIG/java.conf`.

   For instance

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

Installation with Ingres

Introduction

This guide does not claim to precisely describe the installation of the database system. Details about this can be found in the Ingres documentation. Normally, though, this guide should be adequate for performing a ”standard” installation.

Installation

1. Install Ingres

   We assume that the Ingres system will be installed under the user `ingres`. The installation identifier is taken here as being `II`, which is the default value.

2. Create the user `bicsuite`

   There are two ways of registering the user `schedulix` in the Ingres system. The first method involves creating the user with the help of the tool `accessdb`. This method is not explained here any further.

   The second method is to create the user using SQL commands. To do this, start the SQL Terminal Monitor as the user ingres:

```
$ 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. Create the repository database 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. Create and initialise the database tables

Run the following commands to create the required tables:

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

5. Configure the database connection in the schedulix Server configuration file `$BICSUITECONFIG/server.conf`

   Change the following properties as shown here:

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

6. Configure the schedulix Java Class Path for the Ingres JDBC

   Now all you have to do is to append the path to the Ingres JDBC to `CLASSPATH` in the configuration file `$BICSUITECONFIG/java.conf`.

   For instance:

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

Configuration of the TLS/SSL connections

Introduction

Configuring encrypted communication within the schedulix system is relatively easy. However, the required time and effort will vary depending upon your requirements.
The following options are available:

1. schedulix without SSL/TLS communication
2. schedulix with and without SSL/TLS communication
3. schedulix exclusively with SSL/TLS communication

In turn, there are two methods of communicating with TLS/SSL:

1. Server-side authentication only this means that clients can verify whether the server with which they are communicating really is trustworthy. This setting just requires one key pair for the server. All communication is encrypted.
2. Server-side and client-side authentication With this configuration, both parties verify whether the identity of the communication partner is known to them. This setting is extremely secure but also more time-consuming since a key pair has to be generated for each client and obviously for the server as well. All communication is naturally encrypted.

TLS/SSL configuration

Configuring the SSL/TLS communication essentially requires just a few steps.

1. Generate the key pairs for the server

   The utility `keytool`, a component of Java (SE), is used to generate the key pairs. More information about this utility can be found at

   http://download.oracle.com/javase/6/docs/technotes/tools/windows/keytool.html

   The following procedure should work (for non-specialists):

   a) Create a directory, e.g.

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

   b) This directory will afterwards contain the private key for the server and should therefore be protected accordingly, e.g.

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

   c) Generate a key pair using keytool, e.g.

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

   IMPORTANT: Both passwords (keypass and storepass) *must* be identical!

   d) The server's public key must later be given to the clients. To do this, it first has to be extracted from the key file.

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

   The storepass (`secret`) is required here.

2. Modify the server configuration

   The server configuration now has to be modified to notify the server about its keys. You also have to decide whether you want to use just server-side authentication or authentication by both the server and the client. A decision also has to be made with regard to non-encrypted communication.

   ```
   #
   # SSLPort: port for encrypted communication
   # (inactive = 0)
   #
   SSLPort=2507
   #
   # KeyStore defines which key file is to be used
   # $BICSUITEHOME/etc/certs/svrkeystore
   ```

```
#
KeyStore=/home/schedulix/etc/certs/svrkeystore
#
# The KeyStorePassword is required to read out the key store
#
KeyStorePassword=secret
#
# TrustStore defines which file contains the (public) keys of the
# valid communication partners
#
TrustStore=/home/schedulix/etc/certs/svrkeystore
#
# The TrustStorePassword is required to read out the trust store
#
TrustStorePassword=secret
#
# ClientAuthentication defines whether clients have to authenticate
# themselves (true) or not (false)
#
ClientAuthentication=true
#
# Port defines the port for non-encrypted communication
# (inactive = 0)
# If all ports are configured as being inactive,
# port 2506 is opened for non-encrypted communication
#
Port=2506
#
# ServicePort defines the port for accessing the system in
# emergencies (inactive = 0)
#
ServicePort=2505
```

If Client Authentication=true, the server requires both the key store and the trust store. If client authentication is not necessary, only the key store is required.

3. Client configuration

   If no client authentication is necessary, clients only need access to a trust store to be able to verify the server's identity. If the identity of the clients has to be verified as well, a key store will also need to be generated for each client. This is done analogue to the creation of the key store for the server.

4. Modify .sdmshrc

   The use of an "ini" file is mandatory if `sdmsh` or the standard utilities are to communicate with the server via TLS/SSL. There are three possibilities here:

   a) `$BICSUITECONFIG/sdmshrc`

   b) `$HOME/.sdmshrc`

   c) A file specified when the utility is called

The information required for the secure connection is the decisive factor here. For instance:

```
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
```

This permits symmetrical authentication. If only server-side authentication is necessary, the lines concerning the key store can be deleted.

5. Jobserver configuration

   The jobservers now have four new configuration parameters corresponding to the server parameters:

   ```
   KEYSTORE
   TRUSTSTORE
   KEYSTOREPASSWORD
   TRUSTSTOREPASSWORD
   ```

   These are supplemented by another parameter which states whether encrypted communication is desired:

   ```
   USE_SSL
   ```

   If client authentication is to be used, a key pair has to be created for each jobserver. This is done in precisely the same way as has already been described for the schedulix Server.

   You must obviously make sure that the configuration parameter `RepoPort` in particular is set correctly.

6. Getting it all together

   Last, but not least, the public keys now have to be swapped between the individual communication partners using keytool. To define the server's public key as being trustworthy, for instance, it is added to the client's trust store. For example:

   ```
   $ 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
```

A storepass is naturally required to do this.

If client authentication is required as well, the client certificates will also need to be added to the server's trust store.

Installation (Zope4+)

1. Requirements

   To install Zope4 or Zope5, the following packages have to be installed:

   a) python3

   ```
   $ sudo yum install python3
   ```

   b) python3 development headers

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

2. Create a python virtual environment for your Zope installation

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

3. Installation of the Zope5 software

   Install the latest stable release of Zope5. At the time of this writing this is Zope 5.1.2.

   ```
   $ 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
   ```

   If internet access is not available during the installation, Zope can also be installed offline. To do this, proceed as follows:

a) Download python packages

On an identical as possible system with internet access, execute the following commands:

```
$ wget \
https://raw.githubusercontent.com/zopefoundation/Zope/5.1.2/requirements.txt
$ pip download -r requirements.txt -d packages
```

b) Transferring files to the target system

The file 'requirements.txt' and the directory 'packages' now have to be transferred to the target system without internet access. Place the files in the directory $HOME/software.

c) Installation on the target system

The following command installs Zope from the downloaded files:

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

4. Create a Zope instance for schedulix!Web

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

The password can be chosen freely and is used later on, but the user name must be sdmsadm.

Test:

Start the Zope5 instance using the command:

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

The browser should show an 'Zope Auto-generated default page' for the Url http://localhost:8080.

Pressing Strg-C or clsing the terminal window will stop the insance again.

5. Installation of the schedulix!Web components

To install the schedulix!Web components , the Zope installation has to be extended by some modules:

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

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

Now the Zope instance has to be started again to enable access to thise extensions.

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

The Zope Management user interface is now opened at the address

`http://localhost:8080/manage`

in a browser. This is done with the user `sdmsadm` together with the password you have assigned.

The front end software is now loaded into Zope (Import button):

a) Import into the folder / SDMS.zexp.

b) In the folder /SDMS/Install, mark and copy the folders User and Custom.

c) Create the folders User and Custom in the folder / by pasting them.

6. Configure the server connections

   The server connections are also configured in the Zope Management user interface. To do this, log on as the user `sdmsadm`.

   The `SDMSServers` Python script is edited in the folder Custom. This script delivers a dictionary which has to contain an entry for every schedulix Server that is to be addressed by this schedulix!Web installation. The entry looks like this:

```
# Server name that identifies the server in the schedulix
# user interface
'servername' : {
    # IP address or hostname at which the schedulix Server is running
    'HOST'    : 'hostname',

    # Port at which the schedulix Server is addressed
    'PORT'    : '2506',

    # BASIC, PROFESSIONAL, ENTERPRISE
    'VERSION' : 'BASIC',

    # Optional property stating whether the schedulix GUI should
    # cache the server connections
    'CACHE'   : 'Y'

    # Optional property stating for how long cached schedulix GUI
    # server connections should continue to be valid
```

```
    # Default setting is 60 seconds, only significant if  'CACHE' : 'Y'
    'TIMEOUT' : '60'
}
```

An entry with the name `DEFAULT` must exist for bootstrapping. This entry can be deleted after the user has been set up (who obviously should not then use this connection).

If a server is to be addressed via a secure SSL connection, the following additional properties have to be defined as well:

```
# Connection is set up via Secure Socket Layer
'SSL'        : 'true',

# If stated, the identity of the BICsuite! Server is verified
# The stated file must contain the BICsuite!Server server certificate
'TRUSTSTORE' : 'truststore.pem',

# If the BICsuite!Server requires client authentication,
# this property must be defined and the stated file
# must contain the client's certificate and private key.
# The certificate must be known to the server in its trust store.
'KEYSTORE'   : 'keystore.pem'
```

Note:
When using SSL, for performance reasons it is advisable to use cached server connections because setting up a secure connection is a resource-intensive operation.

7. Open the schedulix interface

   The user interface is now available at the address

   ```
   http://localhost:8080/SDMS
   ```

   A logon prompt is displayed when this page is opened. When the user has logged on, the application is started by clicking the "Take Off" button.

   How to work with the interface is described in the relevant documentation.

HTTPS using Zope behind an Apache Web Server

The simplest way to enable HTTPS access to the schedulix!Web Frontend is to use an Apache web server as a proxy in front of Zope.
This section will not go deep into the configuration of apache but describes how to achieve HTTPS communication in an easy way.
Of course the Apache web server and its `mod_ssl` extension has to be installed. The exact process differs slightly for different operating systems and linux distributions.
In a RHEL or CentOS Environment the only thing to do here is:

```
# yum install httpd mod_ssl
```

In and Ubuntu Environment only a

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

is neccessary.
Now the ssl Engine of Apache has to be configured. Often the package manager will do a lot of the job and will create a self-signed certificate which can be used or replaced by an official certificate.
Further on, a proxy rule hast to be defined in the ssl configuration for the virtul host used. In principle, this looks like the following:

```
#
# 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>
```

Please note: For layout reasons some lines have been wrapped which is indicated by a backslash. The actual configuration should contain unwrapped lines without the backslash.
Since the Zope Server is running on the same host as the apache web server, the request will forwarded to `127.0.0.1:8080`. As own hostname `myvirtualhost` is used in this example.
In request header the communication protocol will be set to https.
The only thing left to do, is to redirect any attempt to access the server using http to https. This can be done in the global configuration by adding the following line the configuration of your virtual host:

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

SSO for schedulix with Zope

Introduction

In small environments, the user management integrated in schedulix is practical and allows systems to be easily kept separate. This changes dramatically, however, when the environments become larger. Centralised management is essential to keep track of the different systems, their users and the associated rights.

In many cases, such a management capability is implemented using Microsoft's Active Directory. One interesting function is the single sign-on (SSO) principle. Here, users are authenticated when they log on to their workstations. They do not have to enter their password again when accessing a system that supports SSO. Instead, a token is used to establish that the authentication has already taken place. Apart from the fact that this makes things more convenient for the user, no sensitive data is exchanged in the process either, which makes for significantly greater security.
Using a model environment, this chapter explains how to integrate schedulix with Active Directory. It will make it easy for you to reproduce this in your environment. In our model environment, the schedulix and Zope servers are installed on a CentOS 7 Linux system. The computer is called centos7sso!. The network also contains an Active Directory server called `adserver`, as well as a Windows client. Both Windows machines are in the Windows domain `INDEPENDIT.DE`.

Procedure

To begin with, an Apache web server is installed together with the required modules and configured. The aim is to achieve not only the SSO functionality, but also to enable communication by https. The Apache server will communicate with the Zope server via a `proxy_html` interface. Since the Scheduling Server also performs user authentication, it likewise has to be made aware of the SSO situation.

Kerberos installation and configuration

The SSO protocol internally relies on Kerberos. Therefore the required software hast to be installed and configured. The installation is easy. In a RHEL or CentOS environment a mere

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

will do the job. The file `/etc/krb5.conf` contains the configuration of the Kerberos installation. In our model environment it looks like:

```
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
```

The mentioned file `/etc/httpd/krb5.keytab` can be created on the Active Directory server and deleted after copying it over:

```
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
```

It is possible, that instead of `RC4-HMAC-NT` another encryption algorithm, like e.g. `AES256-SHA1` has to be chosen. From the security perspective, it is recommended to use modern encryption algorithms.
The Apache server requires access to the Active Directory server to be able to perform the authorisation. In our example, we used `bicsuite@INDEPENDIT.DE`, an account without any special privileges. The name itself is insignificant. `VerySecret` was set as the password. A password that is easy to guess, even if it's "VerySecret".
In order to test the Kerberos configuration the following statement will help:

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

The expected output looks like:

```
[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
```

(For layout reasons the output has been shortened and reformatted).

Apache web server and modules

Installation

The next step is to install Apache (httpd) and several required modules. In case of RHEL/CentOS 7 there is a choice between `mod_auth_kerb` and `mod_auth_gssapi`. From RHEL/CentOS 8 on, the Kerberos module isn't supported any longer. Hence the gssapi module is the only option left.
This means either

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

to obtain the Kerberos module, or

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

for the gssapi module.

Configuration

On Red Hat-based systems, Apache is configured under `/etc/httpd` and in some subdirectories. This may be different for other distributions, but the principle remains the same.
First of all, it is ensured that communication with the Apache web server only takes place via https. To do this, some parameters need to be entered in the file `/etc/httpd/conf.d/ssl.conf`. In our model environment these are:

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

Depending on the environment, the intermediate certificates may naturally also play a role.
In the file `etc/httpd/conf/httpd.conf`, it is then ensured that any requests sent to the standard http port are redirected to the https port:

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

The firewall should obviously also be informed as well, for example like this:

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

The user authentication looks a bit more complicated and needs to be supplemented with information from the environment. Depending on the module that is used, one of both similar albeit different configurations apply. In case of `mod_auth_kerb` the following configuration is used in our model environment:

```
# If the AD users belong to multiple groups, the
# FieldSize can quickly become exhausted and you run into errors
# For this reason it is enlarged here
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>
```

The configuration in case of `mod_auth_gssapi` looks similar, even a bit simpler:

```
<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>
```

If the web user requests the location bicsuite, or naturally a resource below the bicsuite folder, Kerberos checks whether the user is authorised to do so. This requires the domain (in our case `INDEPENDIT.DE`) as well as a file `krb5.keytab` which was stored in the Apache configuration directory.
If the authentication is successful, the request is forwarded to the Zope server using http. Any authorisation headers are sent as well.

SELinux

If SELinux is activated, the Apache server is prohibited from opening socket connections itself. However, since this is required for the communication between Apache and Zope, it must be allowed:

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

Results test

Whether everything has gone well so far can be tested. To do this, a directory (e.g. `ssotest`) is created in the DocumentRoot, in this case `/var/www/html`. A file `index.html` is created in the directory with content such as this:

```
<DOCTYPE html>
<html>
    <body>

        <h1>My First Heading</h1>

        <p>Hello World</p>

    </body>
</html>
```

The Apache configuration now has to be temporarily modified for the test. This is done by editing the `location` section as follows:

```
<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>
```

If `https://centos7sso/ssotest` is now accessed from a Windows workstation after the Apache server has been restarted, "Hello World" should be displayed. Calling `wget` from the server itself, such as with `wget https://localhost/ssotest`, should trigger a "401 Unauthorized" page.
If the test was successful, the changes should now be undone again.

Zope extension and configuration

The second player is the Zope server, which should naturally learn about its good fortune as well. Zope must likewise be able to talk to the Active Directory server. To do this, it requires the `python_ldap` package, whose installation in turn requires the `openldap-devel` package:

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

It is advisable to prepare Zope for using SSO before importing `SDMS.zexp`. Although SSO can also be configured later, this requires a few extra steps. In particular, it must be noted that the URL used to address the GUI is different compared to a normal installation.

Product installation

A user ID is required for Zope to be able to execute requests in the right context. Without SSO, the login takes place in Zope, which means that the information is available. With SSO, Zope is sent the user information from the Apache server. An extension is required so that Zope can handle the information.
The installation is easy:

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

It is important to make the port accessible to Apache, but not externally. With the aid of firewalld, that can look like this:

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

Installing SDMS.zexp

The GUI application is installed in a similar way to the installation without SSO. What is important to note here, however, is that the installation takes place in a subfolder instead of the root folder. The name of the folder is not so important as long you always use the correct name.
For technical reasons, two empty folders and another folder for installing the software in the Zope root folder are required. To do this, the Zope management interface is opened by bypassing the Apache server. In our model environment, the URL

```
\verb!http://centos7sso.independit.de:8080/manage!
```

is entered in the browser. If a login prompt is displayed, the user credentials that were entered when installing Zope have to be used. This is typically the user `sdmsadm`.
First of all, the folders `bicsuite`, `web` and `GUI` are now created. A property `SDMSROOT` with the value `/bicsuite/GUI` is also created. Later, the URL

```
\verb!https://centos7sso.independit.de/bicsuite/GUI/SDMS!
```

is used to open the GUI.
Now all the steps follow as with the normal installation, except that SDMS.zexp is imported into the folder GUI!. The folders `Custom` and `User` are also created in the 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: Within the `Custom` folder there is a configuration script called `SDMSServers`, which is used to define which scheduling servers can be accessed by this Zope instance. It is crucial to tell Zope that it should use the SSO protocol.

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

Finally, an object of the type `RemoteUserFolder` is created in the `GUI` folder.

Configuring the SDMS application

Under `$BICSUITEHOME/etc` is a file called `ZopeSSO.conf.template`. This is copied to the configuration directory, which is usually `$BICSUITEHOME/../etc`, and renamed into `ZopeSSO.conf`.
This file is pretty self-explanatory. It begins with general settings, which can then be overwritten for each domain or server.
In our model environment, the following settings were used:

```
# ====================================================================
# 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 Zope 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' : 'G0H0ME',
#-------------------------------------------------------------
# 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
    # overiding 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',
                    #- - - - - - - - - - - - - - - - - - - - - - - - - -
                }
            }
        }
    }
}
```

Configuring the schedulix server

The schedulix server also needs some configuration so that it knows how to respond to SSO logins. What is important is that the settings match the settings for the Zope server. In our model environment, the relevant parameters were set as follows:

```
#
# 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 other 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
```

Settings on the user side

For SSO to function, the browser must know that it should report the credentials to the Apache server. The configuration varies slightly depending on the browser. Which settings need to be configured for some commonly used browsers is explained in the next sections. The order is purely alphabetical.

Chrome and Microsoft Edge

The procedure is the same for both Chrome and Edge. Select the local intranet in the Windows Control Panel under → Network and Internet → Internet Options → Security tab. After clicking the sites and the Advanced button, enter the host name of the Apache server.

Firefox

In Firefox, enter the URL `about:config`. Now search for the word `negotiate`. One of the search results should be

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

Double-click to enter a value. This is the host name of the Apache server. In our model environment this is `centos7sso.independit.de`.
After saving the setting and restarting the browser, the GUI should be accessible without having to enter a password.

Administration of the Zope server

Access separate from the SSO logic is required for administering the Zope server. This is due to the fact that users who are given access via SSO are treated differently from "normal" Zope users.
Nevertheless, the administration should preferably be handled via a secure connection as well. This is possible without any problems, albeit with a few restrictions. Basically, as already explained, the Apache server is configured as a reverse proxy. However, requests which reference `/bicsuite` must not be translated since the SSO logic is supposed to take effect here.
This results in the following addition to the Apache configuration:

```
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"
```

Please note: A line break has been entered for presentation reasons. As usual, this is indicated by a backslash. In the actual configuration, the two lines should be concatenated again without a backslash and whitespace.
This addition can be entered directly after the previously added instructions in `ssl.conf`.
It is advisable to use a different browser from the standard browser for administrative tasks because otherwise some undesirable side effects will occur. This is because you cannot be simultaneously logged in to a Zope instance as two different users in one browser.