independIT Integrative Technologies GmbH
Bergstraße 6
D–86529 Schrobenhausen

schedulix

Installation Guide
Release 2.9

Dieter Stubler Ronald Jeninga

April 10, 2019

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 1.7 JDK or later
  `http://www.oracle.com/technetwork/java/index.html`
  Alternatively including an OpenJDK 1.7 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-2.9.tgz
- Oracle(Sun) Java 1.7 SE JRE
  http://www.oracle.com/technetwork/java/index.html
  Alternatively an OpenJDK 1.7 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-2.9.tgz
- Oracle(Sun) Java 1.7 SE JRE
  http://www.oracle.com/technetwork/java/index.html
  Alternatively an OpenJDK 1.7 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 7.

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-1.7.0-openjdk-1.7.0.0
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 14)
   - MySQL (page 16)
   - Ingres (page 17)

   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-2.9.tgz
   ```

   Create a symbolic link:

```
$ ln -s schedulix-2.9 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 systemâĂŹs 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 17,
   - MySQL, see page 16, and for
   - PostgreSQL, see page 14.

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-2.9.tgz
   ```

   Create a symbolic link:

   ```
   $ ln -s schedulix-2.9 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`.

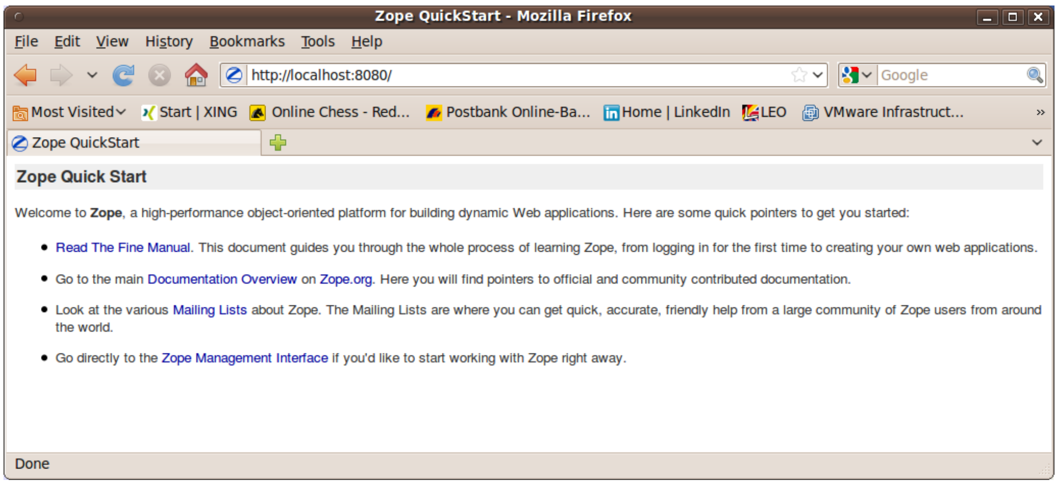
   For instance:

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

Installing the Zope server

Introduction

A Zope Application Server has to be installed before you can use the schedulixinterface.



Zope Quick Start

Welcome to **Zope**, a high-performance object-oriented platform for building dynamic Web applications. Here are some quick pointers to get you started:

- Read The Fine Manual. This document guides you through the whole process of learning Zope, from logging in for the first time to creating your own web applications.
- Go to the main Documentation Overview on Zope.org. Here you will find pointers to official and community contributed documentation.
- Look at the various Mailing Lists about Zope. The Mailing Lists are where you can get quick, accurate, friendly help from a large community of Zope users from around the world.
- Go directly to the Zope Management Interface if you'd like to start working with Zope right away.

Figure 3.1: Zope Quick Start page

Installation

1. Install virtualenv

   ```
   $ easy_install virtualenv
   ```

2. Create a virtual Python environment for the Zope installation

   ```
   $ mkdir $HOME/software
   $ cd $HOME/software
   $ virtualenv --no-site-packages Zope
   ```

3. Install the Zope2 software

   Install the latest release of Zope2. At the time this document was written, 2.13.29 was the most recent release.

   ```
   $ cd $HOME/software/Zope
   $ bin/pip install -r \
   https://raw.githubusercontent.com/zopefoundation/Zope/2.13.29/requirements.txt
   ```

   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/2.13.29/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:

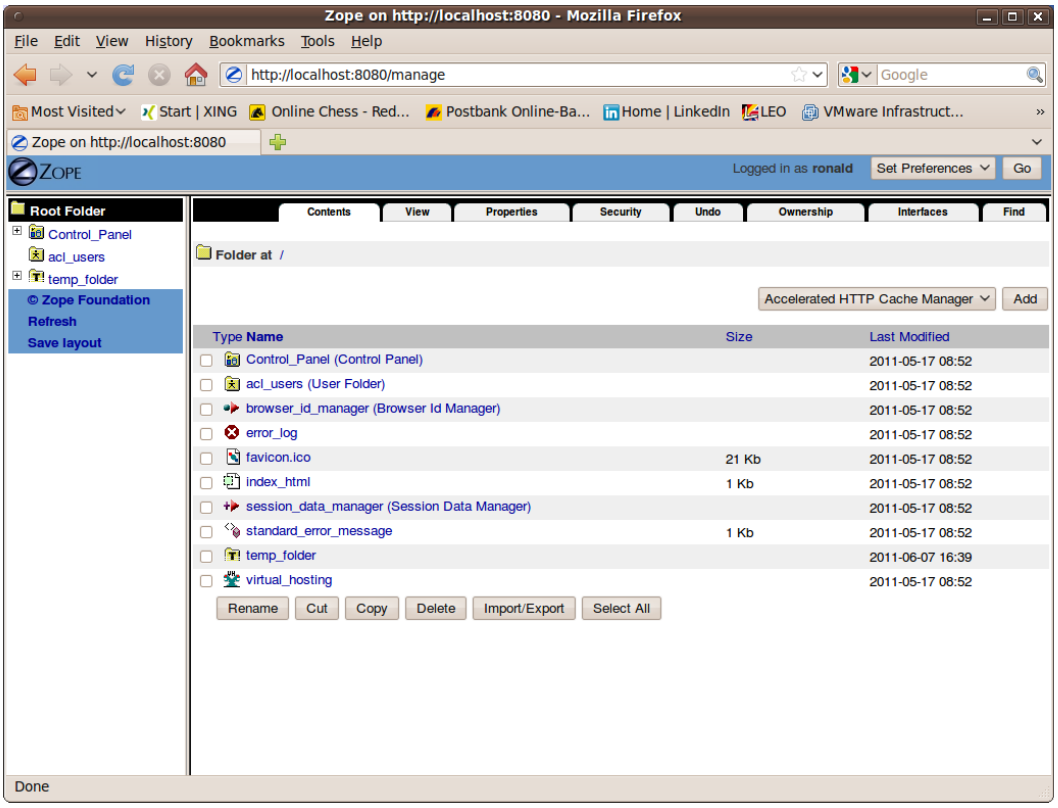
      ```
      $ cd $HOME/software
      $ Zope/bin/pip install --no-index --use-wheel \
      --find-links=./packages  -r requirements.txt
      ```

4. Create a Zope instance for schedulix user interface

   ```
   $ cd $HOME/software/Zope
   $ bin/mkzopeinstance -d $HOME/schedulixweb -u sdmsadm:sdmsadm_password
   ```

   You can use any password of your choice. This will be required again later again. The username must be `sdmsadm` though.

   The Zope server is started briefly for testing:



Zope on http://localhost:8080 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/manage

Logged in as ronald | Set Preferences | Go

Root Folder: Control_Panel, acl_users, temp_folder; © Zope Foundation, Refresh, Save layout

Contents | View | Properties | Security | Undo | Ownership | Interfaces | Find

Folder at /

Accelerated HTTP Cache Manager | Add

Type Name	Size	Last Modified
Control_Panel (Control Panel)		2011-05-17 08:52
acl_users (User Folder)		2011-05-17 08:52
browser_id_manager (Browser Id Manager)		2011-05-17 08:52
error_log		2011-05-17 08:52
favicon.ico	21 Kb	2011-05-17 08:52
index_html	1 Kb	2011-05-17 08:52
session_data_manager (Session Data Manager)		2011-05-17 08:52
standard_error_message	1 Kb	2011-05-17 08:52
temp_folder		2011-06-07 16:39
virtual_hosting		2011-05-17 08:52

Rename | Cut | Copy | Delete | Import/Export | Select All

Done

Figure 3.2: Zope Management screen

```
$ $HOME/schedulixweb/bin/zopectl start
```

In the web browser, the URL

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

should now be displayed on the Zope QuickStart page as in Figure 3.1.

The Zope instance is now stopped again:

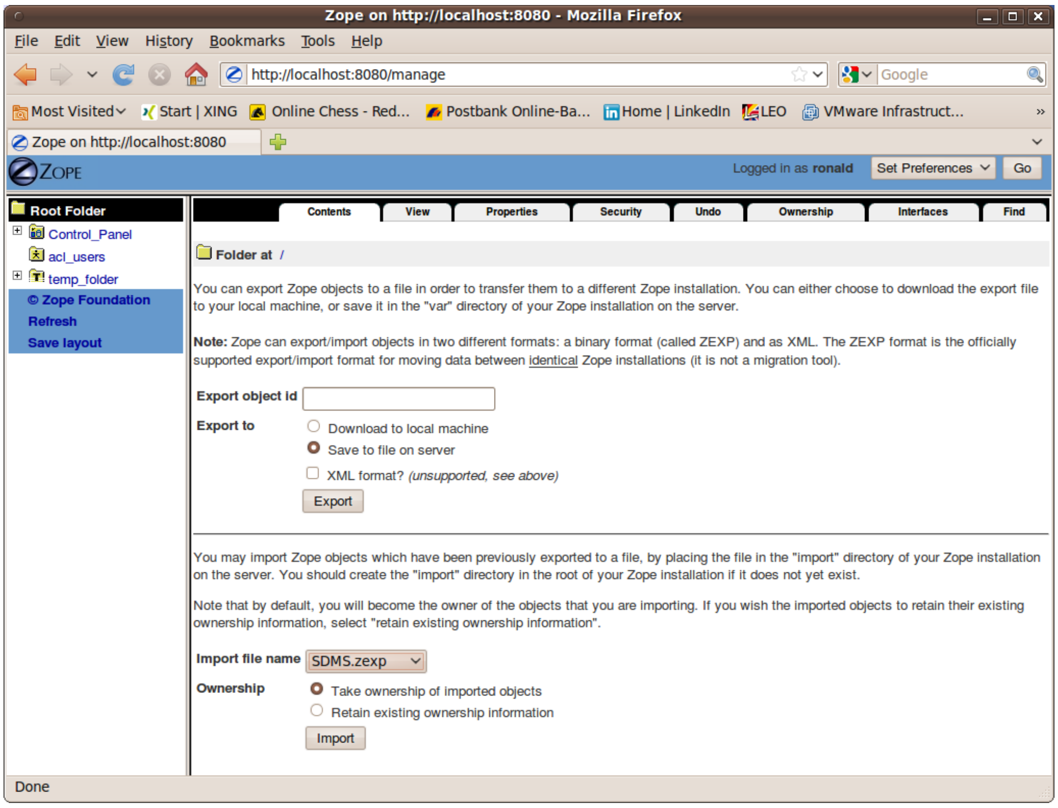
```
$ $HOME/schedulixweb/bin/zopectl stop
```

5. Install the schedulix GUI components

   The Zope installation has to be extended with several modules before the schedulix GUI components can be installed.

```
$ cd $HOME/schedulixweb
$ mkdir Extensions
$ cd Extensions
$ ln -s $HOME/schedulix/zope/*.py .
$ cd ../Products
$ ln -s $HOME/schedulix/zope/BICsuiteSubmitMemory .
$ cd ../import
$ ln -s $HOME/schedulix/zope/SDMS.zexp .
```

The Zope instance now has to be started again to register the changes on the Zope side.



Folder at /

You can export Zope objects to a file in order to transfer them to a different Zope installation. You can either choose to download the export file to your local machine, or save it in the "var" directory of your Zope installation on the server.

Note: Zope can export/import objects in two different formats: a binary format (called ZEXP) and as XML. The ZEXP format is the officially supported export/import format for moving data between identical Zope installations (it is not a migration tool).

Export object id

Export to ○ Download to local machine [✓ Save to file on server]

☐ XML format? *(unsupported, see above)*

Export

You may import Zope objects which have been previously exported to a file, by placing the file in the "import" directory of your Zope installation on the server. You should create the "import" directory in the root of your Zope installation if it does not yet exist.

Note that by default, you will become the owner of the objects that you are importing. If you wish the imported objects to retain their existing ownership information, select "retain existing ownership information".

Import file name SDMS.zexp

Ownership [✓ Take ownership of imported objects] ○ Retain existing ownership information

Import

Figure 3.3: Zope Import dialogue

```
$ $HOME/schedulixweb/bin/zopectl start
```

The Zope Management user interface is now opened at the address

`http://localhost:8080/manage`

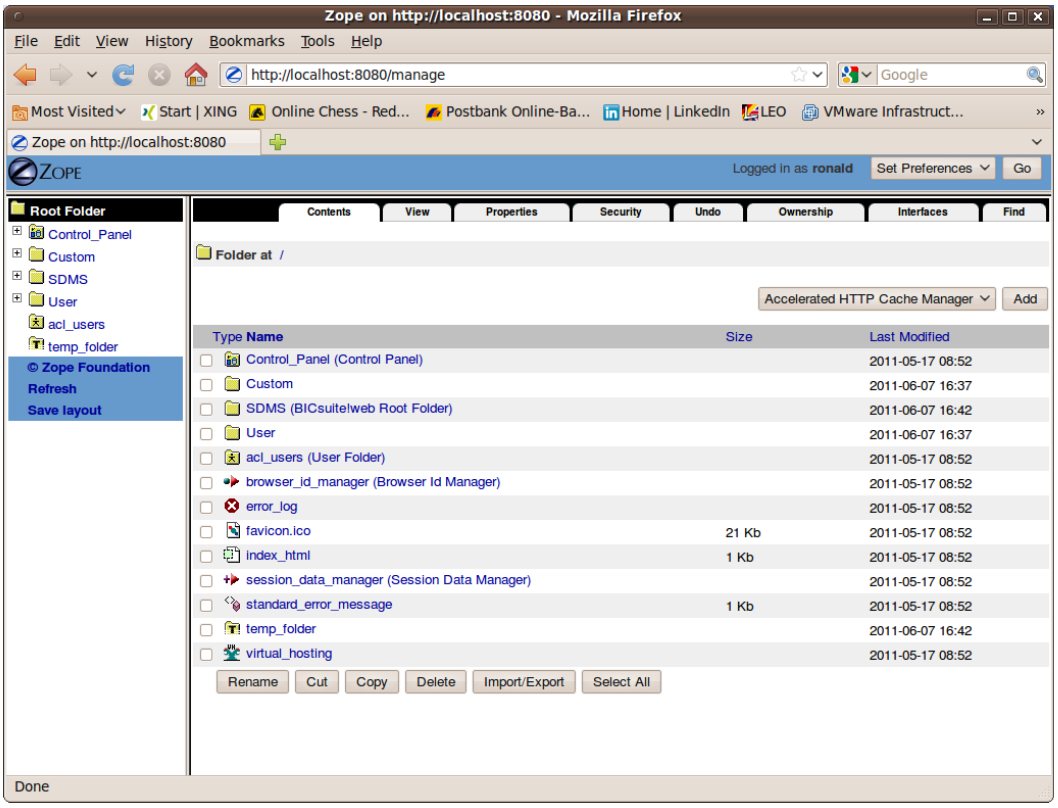
in a browser (see also Figure 3.2). This is done with the user `sdmsadm` together with the password you have assigned.

The front end software is now loaded into Zope (Import button, see Figure 3.3):

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.


Zope on http://localhost:8080 - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://localhost:8080/manage

Most Visited | Start | XING | Online Chess - Red... | Postbank Online-Ba... | Home | LinkedIn | LEO | VMware Infrastruct...

Zope on http://localhost:8080

Logged in as **ronald** | Set Preferences | Go

Root Folder
- Control_Panel
- Custom
- SDMS
- User
- acl_users
- temp_folder

© Zope Foundation
Refresh
Save layout

Contents | View | Properties | Security | Undo | Ownership | Interfaces | Find

Folder at /

Accelerated HTTP Cache Manager | Add

Type Name	Size	Last Modified
Control_Panel (Control Panel)		2011-05-17 08:52
Custom		2011-06-07 16:37
SDMS (BICsuite!web Root Folder)		2011-06-07 16:42
User		2011-06-07 16:37
acl_users (User Folder)		2011-05-17 08:52
browser_id_manager (Browser Id Manager)		2011-05-17 08:52
error_log		2011-05-17 08:52
favicon.ico	21 Kb	2011-05-17 08:52
index_html	1 Kb	2011-05-17 08:52
session_data_manager (Session Data Manager)		2011-05-17 08:52
standard_error_message	1 Kb	2011-05-17 08:52
temp_folder		2011-06-07 16:42
virtual_hosting		2011-05-17 08:52

Rename | Cut | Copy | Delete | Import/Export | Select All

Done

Figure 3.4: Zope result window

If everything has been carried out without any errors, the interface will look like the screenshot in Figure 3.4.

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.

Installing the HTTPS extensions

Installing the Zope HTTPS extension will take some time. It is not particularly complicated, but it is important to carefully and precisely follow the instructions and ideally to understand them as well.

1. Download the `M2Crypto` module

   The M2Crypto module can currently be found at

   http://pypi.python.org/packages/source/M/M2Crypto/M2Crypto-0.21.1.tar.gz

   Unpack it in `$HOME`.

2. Install M2Crypto in your virtual environment

   ```
   $ cd  $HOME/M2Crypto-0.21.1
   $ $HOME/software/Zope/bin/python setup.py install
   ```

   To test whether the installation was completed successfully, you can simply try to load the module in Python:

   ```
   $ $HOME/software/Zope/bin/python
   Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
   [GCC 4.5.2] on linux2
   Type "help", "copyright", "credits" or "license" for more information.
   >>> import M2Crypto
   >>>
   ```

   If this triggers an error message, an `ldconfig` may be required or you may have to close and reopen the terminal.

3. Patch the Zope installation

   The next step ensures that, depending upon the configuration, Zope will actually use the M2Crypto module that has now been installed.

   ```
   $ $BICSUITEHOME/zope/https/patch.sh
   ```

4. Create the files required by Zope HTTPS

   The files required by Zope can be saved to the "etc" directory of the Zope instance, although this is not mandatory. In this guide, it is assumed that this is indeed the case. If you want to adapt this procedure, it is assumed that you have a sound understanding of how both TLS/SSL and HTTPS function.

   First of all, navigate to the ”etc” directory:

   ```
   $ cd $HOME/bicsuiteweb/etc
   ```

   An SSL Certificate Authority is now created:

   ```
   $ openssl req -new -x509 -newkey rsa:2048 -keyout cakey.pem \
   > -out cacert.pem -days 3650
   ```

The following parameters can be entered as an example:

```
Enter PEM pass phrase: super_secret
Verifying - Enter PEM pass phrase: super_secret
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Bavaria
Locality Name (eg, city) []:Schrobenhausen
Organization Name (eg, company) [Internet Widgits Ltd]:independIT
Organizational Unit Name (eg, section) []:Development
Common Name (eg, YOUR name) []:Dieter Stubler
Email Address []:dieter.stubler@independit.de
```

You can now create an SSL server certificate:

- Generate a key for the server certificate:

  ```
  $ openssl genrsa -out serverkey.pem -aes128 2048 -days 3650

  Enter pass phrase for serverkey.pem: dummy
  Verifying - Enter pass phrase for serverkey.pem: dummy
  ```

- Remove the password from serverkey.pem:

  ```
  $ openssl rsa -in serverkey.pem -out serverkey.pem

  Enter pass phrase for serverkey.pem: dummy
  ```

- Generate a Certificate Signing Request:

  ```
  $ openssl req -new -key serverkey.pem -out req.pem -nodes
  ```

  The following parameters can be entered as an example:

  ```
  Country Name (2 letter code) [AU]:DE
  State or Province Name (full name) [Some-State]:Bavaria
  Locality Name (eg, city) []:Schrobenhausen
  Organization Name (eg, company) [Internet Widgits Ltd]:independIT
  Organizational Unit Name (eg, section) []:Development
  Common Name (eg, YOUR name) []:my_hostname
  Email Address []:dieter.stubler@independit.de
  A challenge password []:
  An optional company name []:
  ```

  Important !!!

  `my_hostname` must be replaced with the name that is used by the browser to address the HTTPS host !!!

- Sign the certificate:

  a) Save openssl.cnf (usually requires root privileges)

  ```
  # cp /etc/ssl/openssl.cnf /etc/ssl/openssl.cnf_save
  ```

  b) Edit openssl.cnf:

  ```
  # sudo vi /etc/ssl/openssl.cnf
  ```

  The following entries are modified:

```
dir             = .
private_key     = $dir/cakey.pem
RANDFILE        = $dir/.rand
default_days    = 3650
new_certs_dir   = $dir
```

c) Prepare files:

```
$ touch index.txt
$ echo 01 > serial
```

d) Sign:

```
$ openssl ca -in req.pem -notext -out servercert.pem
```

Example:

```
Enter pass phrase for ./cakey.pem: super_secret
Sign the certificate? [y/n]:y
1 out of 1 certificate requests certified, commit? [y/n]y
```

e) Restore the old openssl.cnf

```
# mv /etc/ssl/openssl.cnf_save /etc/ssl/openssl.cnf
```

- Write the SSL server certificate:

```
$ cat servercert.pem serverkey.pem > server_cert.pem
```

- Create SSL entropy pool :

```
$ dd if=/dev/random of=ssl_entropy_pool.dat bs=1024 count=1
```

- Create ssl_dh_init:

```
$ openssl dhparam -out ssl_dh_init.pem 1024
```

5. Configure Zope HTTPS

If https is to listen at port 8085 and

```
$HOME == /home/bicsuite
```

the following should be added to the file `$HOME/bicsuiteweb/etc/zope.conf` after the closing tag `</http-server>`:

```
<https-server>
 # valid keys are "address", "force-connection-close",
 # required keys are
 #        "x509_remote_users",
 #        "ssl_certificate_authority",
 #        "ssl_server_certificate"
 address 8085
 x509_remote_users off
 ssl_certificate_authority /home/bicsuite/bicsuiteweb/etc/cacert.pem
 ssl_server_certificate /home/bicsuite/bicsuiteweb/etc/server_cert.pem
 ssl_dh_init /home/bicsuite/bicsuiteweb/etc/ssl_dh_init.pem
 ssl_entropy_pool /home/bicsuite/bicsuiteweb/etc/ssl_entropy_pool.dat
 # force-connection-close off
</https-server>
```

If communication should only take place via https, the section `<http-server>` to `</http-server>` is to be commented out.

6. Restart Zope

```
$ $HOME/bicsuiteweb/bin/zopectl stop
```

   We use runzope so that we see any errors straight away:

```
$ $HOME/bicsuiteweb/bin/runzope
```

   `$HOME/bicsuiteweb/log/event.log` is also helpful for analysing problems.

7. Once it has functioned properly, cancel runzope (Ctrl+C) and restart Zope

```
$ $HOME/bicsuiteweb/bin/zopectl start
```

To start BICsuite!Web:
In the browser
http://hostname:8080/SDMS
or
https://hostname:8085/SDMS
The first time BICsuite!Web is started, the server certificate has to be confirmed as being trustworthy in the browser. This is all you need to do in Firefox. In Internet Explorer, however,

/home/bicsuite/bicsuiteweb/etc/cacert.pem

has to be imported under Internet Options → Content → Publishers... → Trusted Root Certification Authorities to prevent warnings from constantly being displayed.