OpenSCADA™ User Guide

Jens Reimann `<jens.reimann@th4-systems.com>`

OpenSCADA™ User Guide

by Jens Reimann

Table of Contents

List of Figures

List of Tables

List of Examples

Chapter 1. Introduction

The following sections give you a short introduction into OpenSCADA™.

1.1. Overview and Purpose

This document describes the different components of OpenSCADA™ from a users point of view.

1.2. Quick Start

This section is intended to give you a really quick start in some of the OpenSCADA™ functionality.

1.2.1. Test Client

There is a test client called `OSTC`. Its purpose is to aid testing and performing generic tasks. It is not a “end user client”.

1.2.1.1. Installation & setup

See section Section 3.1, “Installation and startup” on how to install `OSTC`. Once you have it up and running perform the following steps:

- Start the application
- Switch to the “DA” perspective
- Right click on the “Default File Source” entry in the “Navigator”
- Add two new URIs: `da:net://localhost:1202` and `da:net://localhost:1203`
- Start the two simulation servers from the “Testing” menu [1]
- Right click on the previously created URI entries and issue the “Connect” command

You can now browse through the items provided by the connected DA server.

1.2.1.2. Playing with the memory items

There are some “memory items” which can be used to play around with. Data written to these items will be feed in back as input. These items support both read and write operations.

- Ensure that are switched to the DA perspective and have one real time list and the navigator visible. Also ensure that you have the testing servers running as described earlier.
- Open the connection to `da:net://localhost:1203`
- Navigate to the `memory-cell` folder and right click on the `control` item.
- Issue a write command and write a positive integer number not too big (e.g. 10)
- After the write command there should appear exactly the amount of memory items you requested in your write command
- Drag and drop some of these new items to the real time list. They should appear in the real time list with the ID `memory-cell-X` where X is the number of the memory item used.

- The items in the real time list should have the status “CONNECTED” and a value of “NULL”
- Right click on one item in the real time list and issue another write command. Write any value you like. The value should be provided by the data item after the write command completed

This gives you a short introduction on how to read a write data from a DA server.

1.2.1.3. Some short explanation

In the above examples one might notice that in some cases the label in the navigator (browser) is different than the item id itself. This is ok and correct since the browser can use a different naming and only references the items by id. It can be compared to a file system where the filename is just a name pointing to an inode which identifies the actual file contents. In order to reduce confusion one should try to have some similarity between the browser and the item id.

1.2.2. Setting up an OSGi based master server

After playing around with the client it might be a good idea to try out the “master server”. The master server is normally used to aggregate values from different device drivers and provide them to clients. While aggregating the values they are also processed and archived. Alarms might get generated, formulas and script executed, values changed and security added. The master server is an OSGi™ based application that provides bundles to achieve that.

1.2.2.1. Pre-conditions

You need Eclipse 3.5.2 (at the moment of writing).

You need a checked out version of OpenSCADA or a correctly set up target platform containing all OpenSCADA repositories (see also XXX).

1.2.2.2. Run configuration

Create a new OSGi™ run configuration for Equinox. Activate the following bundles:

- `org.openscada.da.server.osgi`
- `org.openscada.da.server.osgi.exporter.net`
- `org.openscada.da.datasource.item`
- `org.openscada.da.client.net`
- `org.openscada.da.connection.provider`
- `org.openscada.da.client.connection.service`
- `org.openscada.ca.file`
- `org.openscada.ca.servlet`
- `org.openscada.sec.provider.dummy`
- `*jetty*`
- `org.eclipse.equinox.*http`

And add required bundled automatically in order to add required dependencies.

Add the following command line arguments for the Java™ virtual machine: `-Dorg.eclipse.equinox.http.jetty.http.port=8082 -Dopenscada.da.net.exportUri=da:net://0.0.0.0:1204 -Dorg.openscada.ca.file.root=/home/user/cas/test`

Now start the configuration and wait for the OSGi™ prompt.

1.2.2.3. Configure the master server

Now head your web server to `http://localhost:8080/ca` and create the following configuration objects:

- `da.connection` with id `connection.1`
  - `connection.uri=1203`
- `da.datasource.dataitem` with id `datasource.1`
  - `connection.id=connection.1`
  - `item.id=memory-cell-0`
- `da.dataitem.datasource` with any id
  - `datasource.id=datasource.1`
  - `item.id=alias.item.1`

1.2.2.4. Connecting

Start the OSTC and connect to `da:net://localhost:1204`. You should find an item named `alias.item.1` which (when draged to the real time list) is “CONNECTED” but “NULL”.

Start the local test servers from inside OSTC and write (e.g.) 5 to the memory cell control item as described in Section 1.2.1, “Test Client”. Now write some value on the item `memory-cell-0`.

After the write operation is complete you should see that also the item `alias.item.1` has changed to the value you just wrote.

1.2.2.5. Summary

This was just a quick and easy setup of an OSGi™ based server application. Of course there is lots more. See the sections Chapter 6, *Configurable services* for factories that are available and provide more functionality and Chapter 5, *OSGi™ Bundles* which OSGi™ bundles provide the functionality.

The `org.openscada.sec.provider.dummy` bundle is a security provider that allows all access to and does not perform a check of the authentication credentials. This is ok for testing but might be a problem for some deployments.

1.2.3. Quick Start: Exec Driver

The “Exec Server” is an application that provides DA data item based on different shell scripts and command line applications. Processes can be started in different variations and the result (e.g. standard out stream or return code) can be extracted and provided as data items.

This sample gives you a short introduction in setting up a simple sample.

1.2.3.1. Pre-requisites

For running this sample you will need:

- Java 1.5+
- Ant 1.6+
- Subversion 1.5+ for getting the setup

1.2.3.2. Downloading

You need to check out the example project at `http://pubsvn.inavare.net/openscada/modules/examples/trunk/sample_exec1/`. It is an Eclipse project but it can also be checked out on the command line without the need to use Eclipse.

Once you checked out the project, change into the checked out director and start the library download:

```
$ ant download
```

This will start Ant and download the required JAR files.

1.2.3.3. Running

In order to start the exmaple just use Ant and the “run” target:

```
$ ant run
```

The application will start and export the hive on TCP port 1202.

Start up the OSTC and connect to `da:net://localhost:1202` (or which host the application is running on) and you will see the sample data.

1.2.3.4. Configuration

The configuration is located in `configuration/exporter.xml` and can easily be changed using the Eclipse XML editor or any other XML schema aware editor.

Note

You might set the default logging level to “INFO” in the file `configuration/log4j.properties` in order to find errors in the configuration file.

You should also be aware of the fact that the sample configuration uses some commands that might work differently or your computer, produce different output or are completly missing. The “exec server” is only designed to start shell scripts or other command line programs and they might differ from platform to platform or even from version to version.

1.2.3.5. Where to go from here?!

The “exec server” can be quite a powerfull tool. Together with the JDBC server, which is like an exec server for databases, existing stuff can be integrated into OpenSCADA™. The “exec server” also supports extracting the Nagios style return codes so that existing Nagios script can be added to the configuration.

Also the “openscada_ping” application can be added to the exec configuration. This is a C/C++ based application which continously performs ICMP echo/echo reply (aka ping) checks. It extracts the round trip time and package loss and provides continously updated information. Since this requires a raw IP socket you will need the application to have root permissions (e.g. using the SUID bit). You can download a precompiled version and source packages of the application at `http://download.openscada.org/aurora/openscada_ping/`.

Chapter 2. Connection Adapters

The different interfaces of OpenSCADA™ can all be accessed using different connection adapters. This ensures a greater portability across different systems and programming languages. Each functional interface (e.g. Data Access) has a programming language specific specification (interface) and a connection specific specification (protocol specification).

So, for example, the DA interface consists of two interfaces in Java™. They are called `Hive` and `Connection`. The `Hive` interface is the server side and `Connection` the client side[1]. Also there currently exist two protocol specifications for the DA interface. The first is based in the GMPP protocol, which is a proprietary protocol developed by OpenSCADA™. The second is ICE from a company called ZeroC™. Both protocols have the specifics but for both we have a mapping for the OpenSCADA™ DA interface. So in addition to the Java™ interfaces and the protocol specification there are wrappers around the Java™ interfaces which map them to the protocol specification. This allows developers of clients and server to focus on the sole implementation of their application since they don not need to care about protocol issues. On the other side if one wants to connect to OpenSCADA™ using a programming language which is currently not supported by OpenSCADA™ (and I guess there are lots ;-)) he only needs to focus on implementing the protocol specification.

2.1. URI format

OpenSCADA™ uses the known URI format for connection information. The interpretation of the different components of the URI depend on the connection adapter.

The basic format is:

```
interface:connection://username:password@primaryTarget:secondaryTarget/se
```

Where “interface” is the type of interface that the connections describes (e.g. “da” for DA or “ae” for AE). And “connection” is the name of the connection adapter (e.g. “net” for GMPP or “ice” for ICE).

“primaryTarget” and “secondaryTarget” would normally match to hostname (or IP address) and port number.

The query (parameters) part of the URI is used as “connection parameters” or “connection properties”.

In addition, in some cases, the fragment part can be used to add a data item to the URI. This is for example used in the URI format of drag and drop operations of data items.

2.2. GMPP / NET

The GMPP / NET[2] protocol is a plain TCP client/server based protocol.

2.2.1. Connection URI

The connection type identifier is “net”.

[1]The difference between client and server was made in order to let the server aggregate several client connections and, on the other side, reduce the complexity of the client interface.
[2]Do not mix up with .NET

Table 2.1. GMPP connection URI parts

URI Part	Required	Description
Primary Target	required	The hostname of IP address of the server or the network interface to which the server should bind. Can be “0.0.0.0” to bind on any interface.
Secondary Target	required	The number of the port the client should connect to or the server should bind to.
Path	not used	

2.2.2. Connection properties

The following connection properties are supported

			enabled on both server and client! Otherwise communication will not be possible.

Table 2.2. GMPP connection properties

timeout	optional	10000	The default timeout value in milliseconds for all other (more specific) timeouts.
connectTimeout	optional	Defaults to “timeout”	The timeout in milliseconds when establishing a new connection to a server
messageTimeout	optional	Defaults to “timeout”	The default timeout in milliseconds for each message sent using the GMPP protocol. This may be overridden on a message by message basis when passing a message to the protocol stack.
pingDelay	optional	“messageTimeout” divided by “pingFrequency”	The time (in milliseconds) after which a ping packet will be sent if no other packet was received during that time.
pingFrequency	optional	3	The number of packets which will be sent in the time of “messageTimeout” unless overridden by “pingDelay” and unless no other packets arrive.
socketImpl	optional	NIO	The socket implementation to use. See Section 2.2.3, “Socket implementations” for possible values
ssl	optional	false	Indication whether to turn on the SSL engine or not. All following parameters starting with “ssl” will be ignored if SSL is disabled. **Note** This option has to be enabled on both server and client! Otherwise communication will not be possible.
sslProtocol	optional	SSLv3	Specifies the SSL protocol to use. Must be a supported constant from the Java virtual machine that is being used.
sslRandom	optional	*The virtual machine default*	The random number provider used for the SSL context.
sslKeyStoreType	optional	*The virtual machine default*	The password to the key store located at “sslKeyStoreUri”
sslKeyStorePassword	optional	*none*	The password to the key store located at “sslKeyStoreUri”
sslCertPassword	optional	*none*	The password of the certificate to use
sslKeyStoreUri	optional	*none*	The URI to the key store file to use. **Note** If the key store is located in the local file system the prefix `file:/` *has* to be used!

For SSL constant names see *Java™ Cryptography Architecture Standard Algorithm Name Documentation* [http://java.sun.com/javase/6/docs/technotes/guides/security/StandardNames.html] or the equivalent documentation of the virtual machine you are using.

2.2.3. Socket implementations

At the moment the following socket implementations can be used:

NIO Java NIO based implementation.

VM VM internal pseudo-socket implementation.

APR Based on Apache Portable Runtime. Needs `libapr.so` or `apr.dll`

Chapter 3. OpenSCADA Test Client

This section describes the application OpenSCADA Test Client.

3.1. Installation and startup

There are some ways of installing the OpenSCADA Test Client. You can download a compressed archive for your platform, install a Microsoft Windows™ based installer application or build it from source using the Eclipse™ IDE.

3.1.1. Downloading the compressed archive

In order to download the compressed archive version of the OpenSCADA Test Client direct your webbrowser to http://download.openscada.org/orilla/R/0.15.0/ostc for the latest release build or to http://download.openscada.org/orilla/I/0.15.0/ for the latest integration build.

In this directory select the archive file which is appropriate for your platform. For example if you have 64bit Linux™ you will need `ostc-linux.gtk.x86_64.zip`.

Note

For Mac OS X™ there is only one file named `ostc-macosx.carbon.ppc.zip` which contains a “Universal Binary” and is, despite the name, working with PPC and Intel platforms.

After downloading the compressed archive you will need to unpack it and place it somewhere where you remember.

You can start the application by launching the executable `ostc` (or `ostc.exe`, `ostc.app`) using your favorite application launcher.

Note

Be aware that the application will store configuration data inside the applications folder. This means that if you delete that folder your data will be lost.

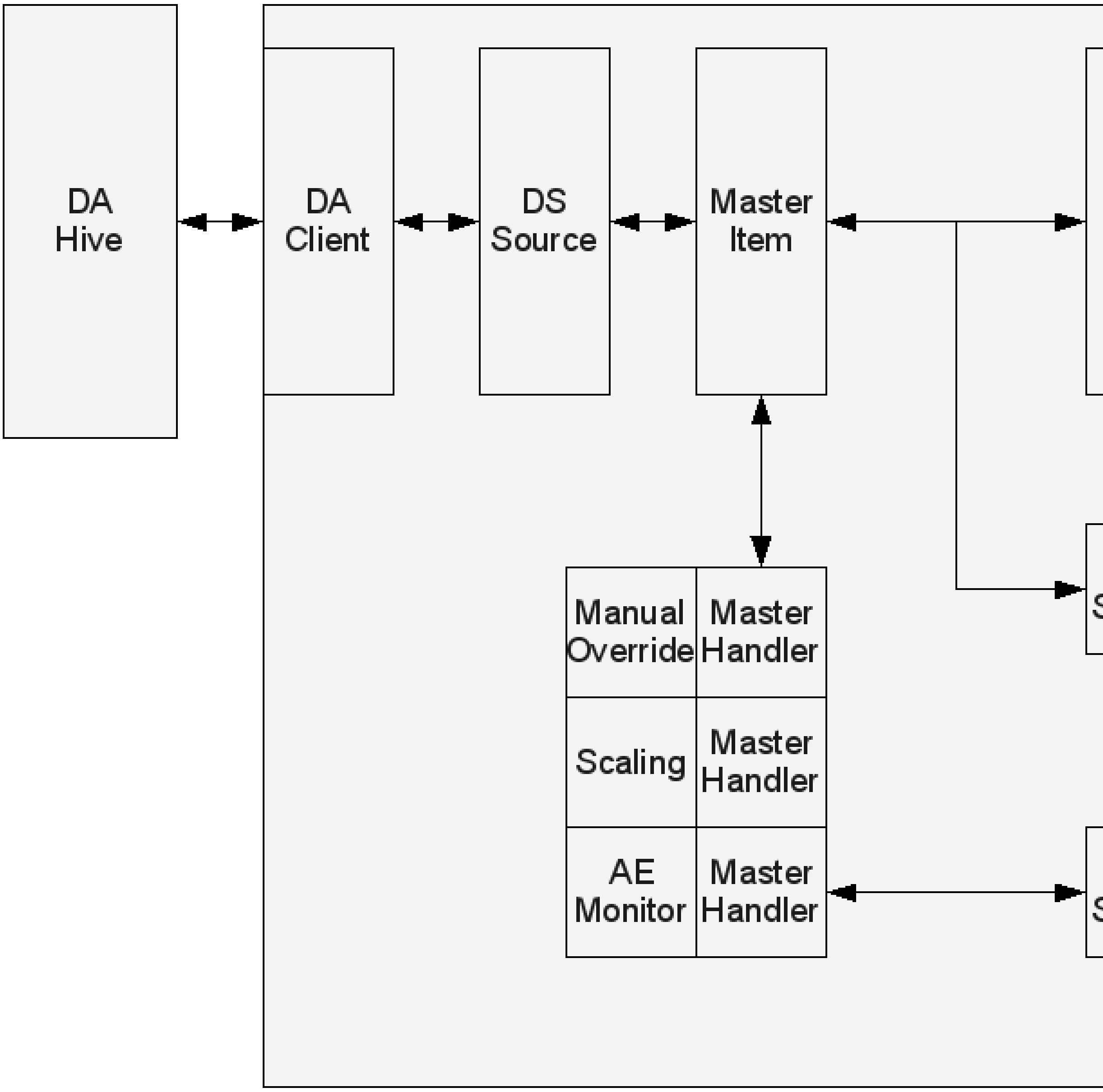
Chapter 4. Master Server

This section describes the “master server” which actually is an OSGi™ container that is configured with several bundles that provide some common SCADA functionality. Using OSGi™ bundled and functionality can be added, removed, updated and reconfigured on the fly as needed. The “master server” itself is a specific set up which contains bundles that provide SCADA functionality.

4.1. Master Server Overview

As seen in illustration Figure 4.1, “Master Server Overview” the master server gets its input from one or more DA hives, processes the data and provides the result using several different interfaces.

Figure 4.1. Master Server Overview



The main processing pipeline is data coming in from the device drivers (“DA hives”), being processes and then provided again using the DA interface. The master server internally uses “data sources” instead of “data items”. Basically the idea is the same, but there are some technical differences needed for the master server in order to better handle data when it is processes inside the master server. So on both ends of the master server there are converters from a data item to a data source and back. On the input side the master server has a connection to a DA hive and, in most cases, several subscriptions to data items. The data provided by these data items is received by the data sources which pass on the information to internal subscribers. At the end of the processing pipeline the last data source will again be converted to a data item so that it can be provided to other DA clients using the standard OpenSCADA™ interfaces.

Beside the source and target data source, explained in the previous paragraph, there can be one or more data sources in between which manipulate data while processing. The most common processing data source is the “master item” which allows to add several “master handlers” the manipulate data in one step. While each data source has its own definition of processing and passing on data the master item will trigger all master handlers at once before passing on information. So a master handler is like a processing data source acquires its input from another data source. The main difference between these two is that the master handler is limited to the trigger of the master item, whereas the data source can delay or aggregate input data as it likes. One the other side a master item handler it much more focused on one data source (the master item) and therefore reduces “data source clutter” in the master server.

Data sources and master handlers can of course alter data provided by the devices or intercept and change write requests. An easy example is the scaling master handler. It simply changes the primary value received from the originating data source and scales it according to actual settings. In addition is extends the secondary values by adding the original source value and the current scaling factor. Write requests are intercepted and specific attribute writes are filtered out and applied to the current settings of the scaling handler.

A second example for a master handler is the level alarm. It checks the primary value and generates a level alarm according to current settings. The result is provided to the A&E system which might generate an alarm condition and log an event. The current alarm condition is then feed back into the secondary values so the further DA clients can read the current level condition by evaluating the secondary values. This might let a process visualization let the value appear in red if it reached a limit.

Since one data source and master handler might depend of results of others the order of processing is important. For data sources this is a matter of configuration. Each data source that processes data gets a source data source configured. This configuration makes up a chain that defines the order of processing. Master item handlers on the other side have a priority. So the master item will call the handlers in that defined order.

4.2. Available data sources

To be written...

4.3. Available master handlers

To be written...

4.4. Setting up a master server

To be written...

4.5. Configuring a master server

To be written...

Chapter 5. OSGi™ Bundles

This section describes the different OSGi™ bundles available in OpenSCADA.

5.1. org.openscada.osgi.equinox.console

The bundle `org.openscada.osgi.equinox.console` provides an implementation of an Equinox™ console that is accessible using a normal TCP (or Telnet) connection.

5.1.1. System properties

The following system properties are available:

Table 5.1. System properties of `org.openscada.osgi.equinox.console`

Property	Type	Required	Description
org.openscada.osgi.equinox.console.port	Integer	optional	The port number the console will bind to. It is important that only one console at a time can bind to a port. Different instances with different consoles must have different ports otherwise the port cannot be opened and the console will not be accessible.
org.openscada.osgi.equinox.console.timeout.login	Integer	optional	The timeout in milliseconds during the login phase (before the user is authenticated). If no input is received for the specified amount of time the connection is closed by the server.
org.openscada.osgi.equinox.console.secret	String	optional	The “secret” (aka “password”) the user has to enter in order to gain access.
org.openscada.osgi.equinox.console.timeout	Integer	optional	The timeout in milliseconds after the login phase (after the user is authenticated). If no input is received for the specified amount of time the connection is closed by the server.

5.2. org.openscada.ds.storage.file

The bundle `org.openscada.ds.storage.file` provides an implementation of the DS (Data Store) service based on the file system.

5.2.1. System properties

The following system properties are available:

Table 5.2. System properties of `org.openscada.ds.storage.file`

Property	Type	Required	Description
org.openscada.ds.storage.file.root	String	Required	The base path to the file system storage where all data will be stored. Read and write access is required. If the directory does not exists it will be created including all parent directories. The path must not point to a file.

5.3. org.openscada.ca.jdbc

The bundle `org.openscada.ca.jdbc` provides and implementation of the CA (Configuration Administrator) service based on a JDBC data source.

This bundle will, most likely, need a fragment bundle that contains the driver class itself or better a reference to the bundle which holds the driver class. Either bundle or package reference are possible.

5.3.1. System properties

The following system properties are available:

Table 5.3. System properties of `org.openscada.ca.jdbc`

Property	Type	Required	Description
org.openscada.ca.jdbc.schema	String	Optional	Name of the schema in the database.
org.openscada.ca.jdbc.instance	String	Optional	Instance name used to differ between different instances using the same table. Defaults to `default`.
org.openscada.ca.jdbc.url	String	Required	URL to the database.
org.openscada.ca.jdbc.username	String	Optional	The user name used when connecting with the database.
org.openscada.ds.storage.jdbc.password	String	Optional	The password used when connecting with the database.
org.openscada.ca.jdbc.password	String	Required	The password for the user.
org.openscada.ca.jdbc.chunksize	Integer	Optional	If the value is greate than zero the data will be split up in chunks instead of put in one record. This is sometimes necessary for databases that have no support for text types of unlimited size. If the field CA_VALUE is limited use this limit as chunk size. Defaults to zero (deactivated).
org.openscada.ca.jdbc.table	String	Optional	The table name to use. Defaults to `ca_data`.
org.openscada.ca.jdbc.fixNull	Boolean	Optional	A flag that indicates whether the database stores empty string as `null` which has to be handled differently in that case. (e.g. Oracle requires `true` here). Defaults to `false`.

5.3.2. Table structure

The following tables have to be created for the bundle using a chunksize greater than zero:

```
CREATE TABLE OPENSCADA_CA (
 INSTANCE_ID VARCHAR(255),
 FACTORY_ID VARCHAR(255),
 CONFIGURATION_ID VARCHAR(255),
```

```
 CA_KEY VARCHAR(512),
 CA_VALUE VARCHAR(4000),
 CHUNK_SEQ INTEGER,

 PRIMARY KEY (INSTANCE_ID, FACTORY_ID, CONFIGURATION_ID, CA_KEY, CHUNK_SEQ)
);
```

Else use the following table structure:

```
CREATE TABLE OPENSCADA_CA (
 INSTANCE_ID VARCHAR(255),
 FACTORY_ID VARCHAR(255),
 CONFIGURATION_ID VARCHAR(255),
 CA_KEY VARCHAR(512),
 CA_VALUE TEXT,

 PRIMARY KEY (INSTANCE_ID, FACTORY_ID, CONFIGURATION_ID, CA_KEY, CHUNK_SEQ)
);
```

5.4. org.openscada.ds.storage.jdbc

The bundle `org.openscada.ds.storage.jdbc` provides an implementation of the DS (Data Store) service based on a JDBC data source.

This bundle will, most likely, need a fragment bundle that contains the driver class itself or better a reference to the bundle which holds the driver class. Either bundle or package reference are possible.

5.4.1. System properties

The following system properties are available:

Table 5.4. System properties of `org.openscada.ds.storage.jdbc`

Property	Type	Required	Description
org.openscada.ds.storage.jdbc.schema	String	Optional	Name of the schema in the database.
org.openscada.ds.storage.jdbc.table	String	Optional	The name of the database table to use. Defaults to `DATASTORE`.
org.openscada.ds.storage.jdbc.instance	String	Optional	Instance name used to differ between different instances using the same table. Defaults to `default`.
org.openscada.ds.jdbc.url	String	Required	URL to the database.
org.openscada.ds.storage.jdbc.username	String	Optional	The user name used when connecting with the database.
org.openscada.ds.storage.jdbc.password	String	Optional	The password used when connecting with the database.
org.openscada.ds.storage.jdbc.driver	String	Optional	The name of the JDBC driver class. Normally this is set be the fragment bundle and must not be specified.
org.openscada.ds.storage.jdbc.encoder	String	Optional	Defines the way binary data is stored. Either `blob` can be used if the field DATA is a BLOB type field. Or `base64` if the binary data is Base 64 encoded. If the base64 option is used the property `org.openscada.ds.storage.j` can also to be specified. Defaults to `blob`.
org.openscada.ds.storage.jdbc.chunkSize	Integer	Optional	If the binary data is base 64 encoded this defines a maximum chunk size the content is split up for databases where the data field is limited.

5.4.2. Table structure

The following tables have to be created for the bundle:

```
CREATE TABLE OPENSCADA_DS (
 INSTANCE_ID VARCHAR(255),
 NODE_ID VARCHAR(512),
 DATA  VARCHAR(4000),
 SEQUENCE_NR INTEGER,

 PRIMARY KEY (INSTANCE_ID, NODE_ID, SEQUENCE_NR)
);
```

5.5. org.openscada.ae.server.storage.jdbc

The bundle `org.openscada.ae.server.storage.jdbc` provides an implementation of the AE (Alarms & Events) service based on a JDBC data source.

This bundle will, most likely, need a fragment bundle that contains the driver class itself or better a reference to the bundle which holds the driver class. Either bundle or package reference are possible.

5.5.1. System properties

The following system properties are available:

Table 5.5. System properties of `org.openscada.ds.storage.jdbc`

Property	Type	Required	Description
org.openscada.ae.server.storage.jdbc.schema	String	Optional	Name of the schema in the database.
org.openscada.ae.server.storage.jdbc.url	String	Required	URL to the database.
org.openscada.ae.server.storage.jdbc.username	String	Optional	The user name used when connecting with the database.
org.openscada.ae.server.storage.jdbc.password	String	Optional	The password used when connecting with the database.

5.5.2. Table structure

The following tables have to be created for the bundle:

```
CREATE TABLE OPENSCADA_AE_EVENTS (
   ID CHAR(36),
   SOURCE_TIMESTAMP TIMESTAMP,
   ENTRY_TIMESTAMP TIMESTAMP,
   INSTANCE_ID VARCHAR(32),
   MONITOR_TYPE VARCHAR(32),
   EVENT_TYPE VARCHAR(32),
   VALUE_TYPE VARCHAR(32),
   VALUE_STRING VARCHAR(4000),
   VALUE_INTEGER BIGINT,
   VALUE_DOUBLE FLOAT,
   MESSAGE VARCHAR(4000),
   MESSAGE_CODE VARCHAR(255),
```

```
   PRIORITY SMALLINT,
   SOURCE  VARCHAR(255),
   ACTOR_NAME VARCHAR(128),
   ACTOR_TYPE VARCHAR(32),

   PRIMARY KEY (ID)
);

CREATE TABLE OPENSCADA_AE_EVENTS_ATTR (
   ID CHAR(36),
   "KEY" VARCHAR(64),
   VALUE_TYPE VARCHAR(32),
   VALUE_STRING VARCHAR(4000),
   VALUE_INTEGER BIGINT,
   VALUE_DOUBLE FLOAT,

   PRIMARY KEY (ID, "KEY"),
   FOREIGN KEY (ID) REFERENCES OPENSCADA_AE_EVENTS (ID) ON DELETE CASCADE
);

CREATE INDEX idx_openscada_ae_1 ON OPENSCADA_AE_EVENTS (SOURCE_TIMESTAMP);
CREATE INDEX idx_openscada_ae_2 ON OPENSCADA_AE_EVENTS (ENTRY_TIMESTAMP);
CREATE INDEX idx_openscada_ae_3 ON OPENSCADA_AE_EVENTS (INSTANCE_ID);
CREATE INDEX idx_openscada_ae_4 ON OPENSCADA_AE_EVENTS (SOURCE_TIMESTAMP,INS
CREATE INDEX idx_openscada_ae_5 ON OPENSCADA_AE_EVENTS (ENTRY_TIMESTAMP,INST
```

5.6. org.openscada.sec.provider.script

The bundle `org.openscada.sec.provider.script` contains and implementation of a authorization provider which is configured using JavaScript or other Java scripting compatible with JSR-223.

5.6.1. System properties

No system properties are used by the bundle.

5.6.2. Exported services

The following services are exported by this bundle:

Table 5.6. Exported services of `org.openscada.sec.provider.script`

Supported interfaces	Description
org.openscada.sec.AuthorizationService	A script language based authorization provider that evaluates requests by iterating through its configured entries as configured using the factory `org.openscada.sec.provider.script.factory`.

5.6.3. Configurable services

The following configurable services are provided by this bundle:

Table 5.7. Configurable services of `org.openscada.sec.provider.script`

Factory ID	Description
org.openscada.sec.provider.script.factory	A configuration entry used to configure the authorization provider. See also Section 6.1, “org.openscada.sec.provider.script.factory”

5.7. org.openscada.sec.provider.dummy

The bundle `org.openscada.sec.provider.dummy` contains and implementation of a authorization and authentication provider which grants access to all operations and authorization credentials.

The services are registered with lowest possible priority in OSGi™ so any other service should override the dummy implementation.

5.7.1. System properties

No system properties are used by the bundle.

5.7.2. Exported services

The following services are exported by this bundle:

Table 5.8. Exported services of `org.openscada.sec.provider.script`

Supported interfaces	Description
org.openscada.sec.AuthorizationService	A dummy authorization service implementation.
org.openscada.sec.AuthenticationService	A dummy authentication service implementation.

5.8. org.openscada.da.server.osgi.summary

The bundle `org.openscada.da.server.osgi.summary` provides summary information summarized over all available data sources.

5.8.1. System properties

No system properties are used by the bundle.

5.8.2. Exported services

The following services are exported by this bundle:

Table 5.9. Exported services of `org.openscada.da.server.osgi.summary`

Supported interfaces	Description
org.openscada.da.datasource.DataSource	Each configured service implements the `DataSource` interface.

5.8.3. Configurable services

The following configurable services are provided by this bundle:

Table 5.10. Configurable services of `org.openscada.da.server.osgi.summary`

Factory ID	Description
org.openscada.da.server.osgi.summary.attribute	A datasource which aggregates all datasources and summarizes by the specifies attribute name. See also Section 6.5, "org.openscada.da.server.osgi.summary.attribute"

5.9. org.openscada.da.server.osgi.exporter.net

The bundle `org.openscada.da.server.osgi.exporter.net` exports the first Hive instance registered with OSGi™.

5.9.1. System properties

The following system properties are available:

Table 5.11. System properties of `org.openscada.osgi.equinox.console`

Property	Type	Required	Description
openscada.da.net.exportUri	URI	optional	The exporter URI that describes the options when exporting the service. The URI must start with "da:net:" since this bundle is specifically a DA NET exporter. The default value is `da:net://0.0.0.0:1202` See Chapter 2, *Connection Adapters* for more information information about possible values.

5.9.2. Exported services

No services are exported by this bundle.

Chapter 6. Configurable services

This section describes the different configurable services of the OpenSCADA™ OSGi™ based system.

6.1. org.openscada.sec.provider.script.factory

This configuration factories creates entries that configure the script based authentication service. Each entry is used in evaluating the authorization request in the case the script based authorization provider is used. The entries will not cause further OSGi™ services to be created but only create internal objects for the service itself.

6.1.1. Configuration

The following configuration properties are available:

Table 6.1. Configuration properties of `org.openscada.sec.provider.script.factory`

			use. This must be a valid and registered JSR-223 script engine. If it is not specified the JavaScript engine will be used.
priority	Integer	optional	The priority of this entry. Gives a sort order in which the entries will be evaluated. Lower number will be evaluates before higher numbers.
for.id	String	optional	A regular expression that will filter out object id for this entry. If the regular expression matches the entry it will be evaluated. Otherwise the entry will not be evaluated for this request. If the property is not set it will be considered as a match.
for.type	String	optional	A regular expression that will filter out object type for this entry. If the regular expression matches the entry it will be evaluated. Otherwise the entry will not be evaluated for this request. If the property is not set it will be considered as a match.
for.action	String	optional	A regular expression that will filter out action for this entry. If the regular expression matches the entry it will be evaluated. Otherwise the entry will not be evaluated for this request. If the property is not set it will be considered as a match.
script	String	required	The script that will be executed. See Section 6.1.2, “Writing authorization scripts” for more information about writing authorization scripts.

6.1.2. Writing authorization scripts

The script has to evaluate if the request provided can be granted, is rejected or if the script cannot decide about that. The script has to provide a return value. How to do that depends on the scripting language you are using. The following samples are based on the JavaScript script engine.

The script fragment will have the following global variables bound:

id	The object id for which the authorization is requested. This variable can be `null`.
type	The object type for which the authorization is requested. This variable can be `null`.
action	The action for which the authorization is requested. This variable can be `null`.
user	The information about the user requesting the authorization. This variable can be `null`.
context	A map containing additional information in key/value pairs. This variable is never `null` but can be empty.
GRANTED	A pre-defined object instance which reflects a “granted” result. This variable is never `null`.

The return value of the script fragment defined the result of the authorization check. If no return value is given or if the return value is `null` then the entry has not voted and the next entry is checked. The application itself decides what will happen if no entry has voted.

The return value can be either the predefined global variable `GRANTED`. It can be a boolean value were `true` means *granted* and `false` means *rejected*. It can be a number where *zero* means *granted* and any other value means *rejected*. The number itself is used as error code. A string where an empty string means *granted* and any other string will mean *rejected*. The string itself is used as error message. It can be an instance of `org.openscada.utils.statuscodes.StatusCode` which always means *rejected*. The status code is used as error code and message. It can be a `Throwable` which always means *rejected*. If it is a status coded exception the status code information will be used as well. It can be an instance of `org.openscada.sec.provider.script.Result` which is a complex result structure that contains all information. If the result type is something else then the previously describes types the request is *rejected*.

The following, rather simple example, shows a method to grant everything. Combined with the `for.id`, `for.type` and `for.action` filters you can have a rather simple start.

Example 6.1. `SampleClient1`

```
GRANTED;
```

6.2. da.connection

This factory creates and registers DA connections. The persistent id of the service is the id of the configuration. By default connections will be automatically re-connected when the connection drops.

Note

The factory cannot create the connections itself since it needs a connection provider that implements the actual connection type. Once the connection provider becomes available in OSGi™ the connection is created and registered.

6.2.1. Configuration

The following configuration properties are available:

Table 6.2. Configuration properties of `org.openscada.sec.provider.script.factory`

Property	Type	Required	Description
connection.uri	String	required	The URI of the DA connection to create

6.2.2. Additional information

For realizing DA NET connections start the bundles `org.openscada.da.connection.provider` and `org.openscada.da.client.net`

6.3. da.dataitem.datasource

This factory creates and registers DA data items with an DA data item object pool. The values this data item provides comes from a defined data source.

6.3.1. Configuration

The following configuration properties are available:

Table 6.3. Configuration properties of `org.openscada.sec.provider.script.factory`

Property	Type	Required	Description
datasource.id	String	required	The id of the source data item which provides the values exported by this data item.
item.id	String	required	The id of the data item when it is exported. This id must be unique.

6.3.2. Additional information

The created data item will subscribe to a data source with configured id (datasource.id). Once it is attached to the data source it will export this data as an data item.

Although the data item is registered using an object pool there are two steps needed in order to access it using a client application (e.g. OSTC) using the DA NET protocol. First a Hive implementation is needed that catches up the data items from the object pools and gathers them in a Hive instance. Next the hive instance must be exported using a protocol implementation.

A ready to run hive implementation is provided by the bundle `org.openscada.da.server.osgi` and a DA NET exporter is available in the bundle `org.openscada.da.server.osgi.exporter.net`. Starting these two bundles will do the trick. Be aware that the exporter bundle exports the service on TCP port 1202 by default. You might need to change that.

6.4. da.datasource.dataitem

This factory creates and registers a data source that gets its input from a client side data item.

6.4.1. Configuration

The following configuration properties are available:

Table 6.4. Configuration properties of `org.openscada.sec.provider.script.factory`

Property	Type	Required	Description
connection.id	String	required	The service id of a DA connection which has the connection to the hive of the data item
item.id	String	required	The item id of the data item to subscribe to in the remote hive

6.4.2. Additional information

The datasource creates a new data item subscription on the specified hive connection. Once the connection is available and established it will automatically subscribe and provide the values to data sources subscribe to itself.

6.5. org.openscada.da.server.osgi.summary.attribut

This factory created and registers data sources which summarize all other data sources by attribute name.

6.5.1. Configuration

The following configuration properties are available:

Table 6.5. Configuration properties of `org.openscada.da.server.osgi.summary.attribute`

Property	Type	Required	Description
attribute	String	required	The name of the attribute by which the summarization should be performed.

6.5.2. Additional information

The datasource creates a new data item subscription on the specified hive connection. Once the connection is available and established it will automatically subscribe and provide the values to data sources subscribe to itself.

Chapter 7. Administration

This section describes some administrative tasks of the OpenSCADA™ system.

7.1. Administration of CA based servers

The CA provides a method of configuration for OpenSCADA™ servers and is commonly used with OSGi™ based applications. OSGi™ based applications make use of an extremely modularized service infrastructure and the CA is the system to configure and manage these service instances.

The CA can be seen as a database which lives inside each main application container (e.g. OSGi™ container) and to which all providers (factories) of services subscribe to. The service factories will receive initial configuration information and following configuration updates from the CA and adapt their provides services to these configuration information. Of course the CA needs to persistently store the configuration data. Depending on the implementation of the CA the data might be stored in a local file system or a database accessed using JDBC.

At the moment there are two implementation of the CA in OpenSCADA™ Atlantis. One is file based (OSGi™ bundle `org.openscada.ca.file`) and one is JDBC based (OSGi™ bundle `org.openscada.ca.jdbc`).

Note

Only one implementation should be active at one time in one application. Otherwise multiple CA running instances will cause unspecified behavior.

Independent of which implementation is active, the CA provides common interfacing methods to other services. Two services of interest are the servlet configurator and the JAX-WS WebService interface. Both can be used to configure the CA from outside the application and will be explained in the following sections.

For a list of services to configure using the CA see Chapter 6, *Configurable services*.

7.1.1. Servlet configurator

The servlet configurator provides an easy web access for human end users using an HTML based interface. The user can access the web page using a standard web browser and create, view, update or delete configuration entries.

Note

The servlet configurator currently shows the whole configuration at once using one web page. If you have a rather huge configuration this might bring your web browser into trouble rendering such a big web page.

7.1.1.1. Activating the servlet configurator

In order to activate the servlet configurator you will need to install and start the following bundles assuming that Equinox™ is used as OSGi™ container:

- `org.mortbay.jetty.server`
- `org.eclipse.equinox.http.jetty`
- `org.openscada.ca.servlet`

In order to choose a different port for the HTTP server than port 80 the system property `org.eclipse.equinox.http.jetty.http.port` has to be set to the specific port number that should be used instead (e.g. 8080).

If a different container than Equinox™ is used the container specific HTTP registry has to be started and then the bundle `org.openscada.ca.servlet`.

7.1.1.2. Accessing the servlet configurator

Navigate to `http://hostname:port/ca` (e.g. `http://localhost:8080/ca`) and a list of all factories and configuration entries will be provided.

7.1.2. Web service interface

The web service interface provides a more machine based access to the CA system. A web service based on JAX-WS is provided which uses HTTP as transport layer but does not provide a human end user interface. Normally an application will access this interface providing its own, custom interface.

The OpenSCADA™ Administration client provides such an interface and allows the user to view, import and export configuration archives. The basic workflow for mass configuration is that a all configuration fragments (like spreadsheet based IO lists and other sources) are compiled to an " OpenSCADA™ Configuration Archive" (OSCAR). This file contains all informations needed and can be imported to the server using the administration client. During the import process the client will read the OSCAR file and the current configuration from the server. The difference between these two is calculated and the required actions are then sent to the server for processing.

Also can the current configuration of a CA be exported from a running server into an OSCAR file for backup, transfer to another server, etc..

7.1.2.1. Activating the web service interface

In order to activate the s web service interface you will need to install and start the following bundles:

- `org.openscada.utils.osgi.jaxws`
- `org.openscada.ca.servlet.jaxws`

The communication endpoint to which the service will be bound can be specificed using the system property `org.openscada.utils.osgi.jaxws.baseAddress`. The value must be a parsable HTTP URI which can also be the "any interface" (0.0.0.0). The value `http://0.0.0.0:9091` will bind the web service to all network interfaces and TCP port 9091.

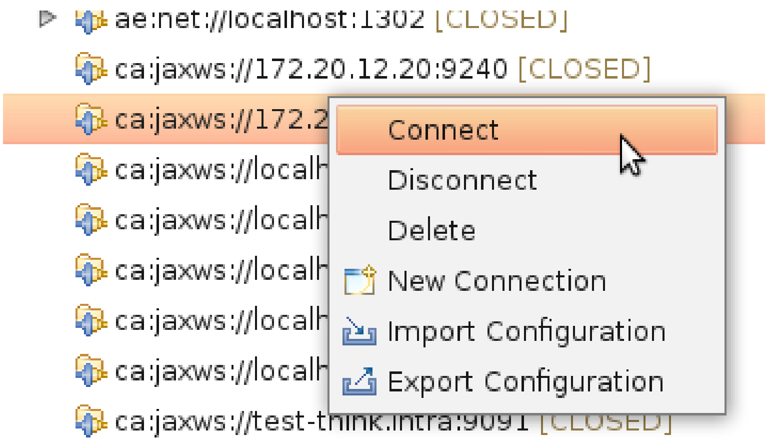
7.1.2.2. Creating a new connection entry

In order to get access from the OpenSCADA™ Administration Client to the CA you will need to add a new connection to the list of available connections. This is just another connection like all other connections in the administration client.

The URI of the a CA connection is `ca:jaxws://hostname:port` (e.g.: `ca:jaxws://localhost:9091`).

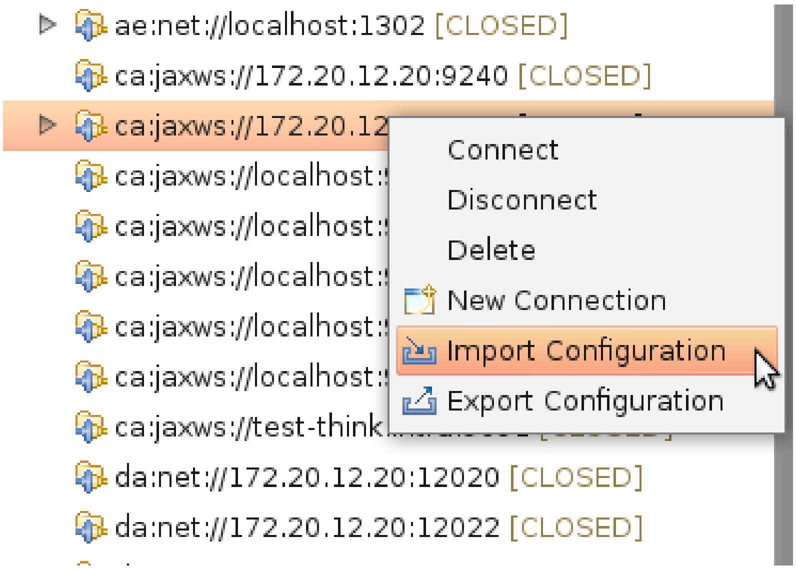
7.1.2.3. Importing a configuration archive

In the "Connections" view of the OpenSCADA™ Administration Client establish a connection to the target CA connection:



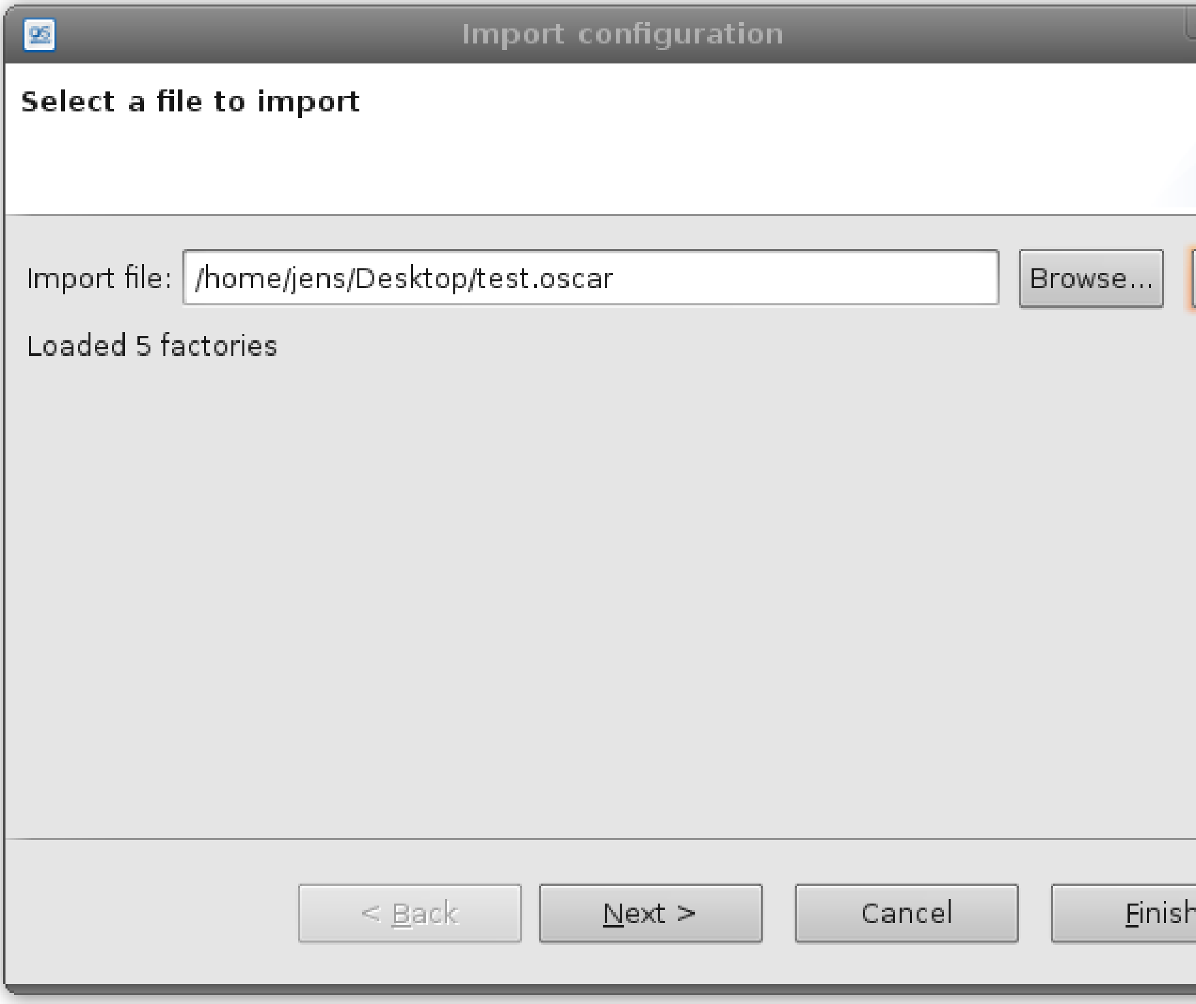
Connect to CA

Once the connection is established use the menu entry Import configuration from the context menu:



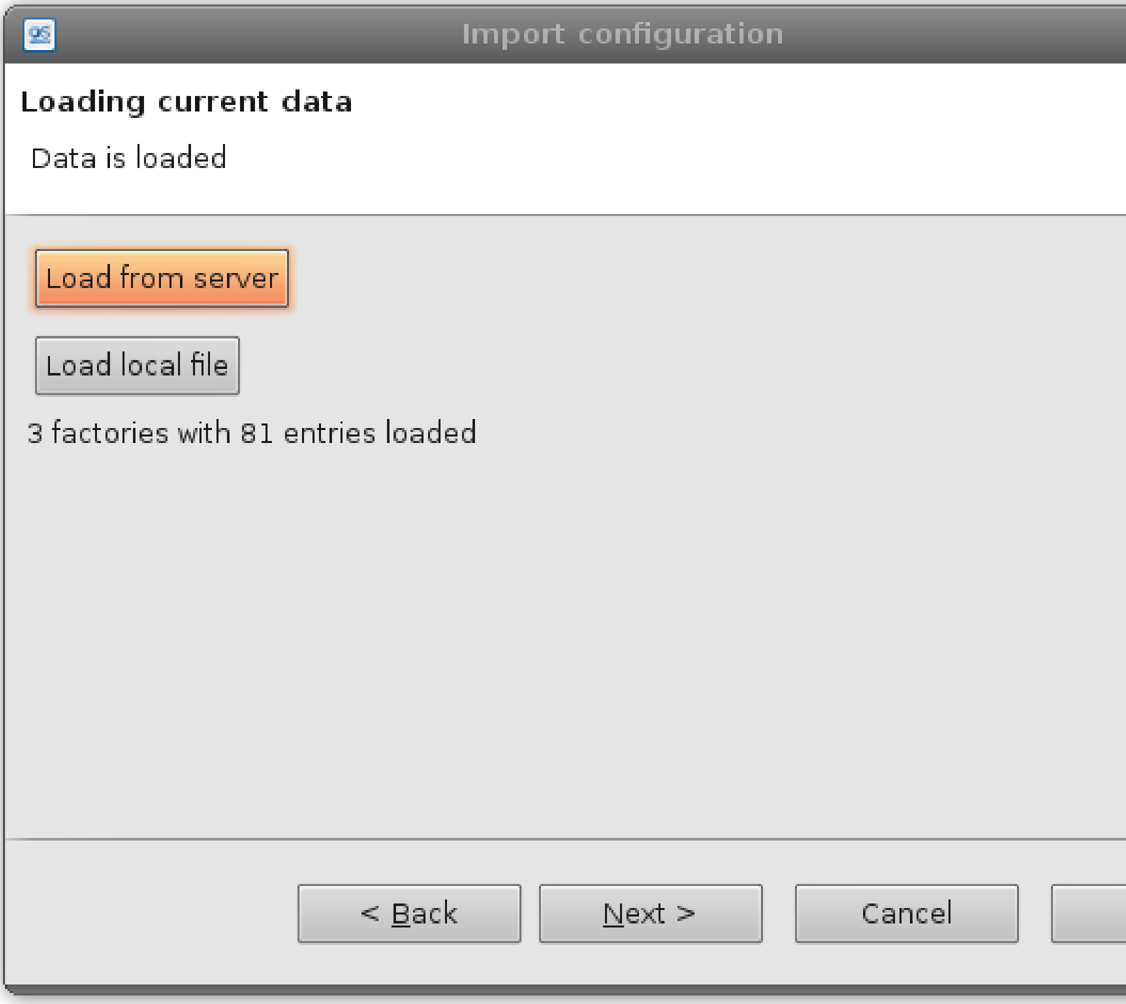
Start import wizard

The import wizard will be opened and required the user to select the configuration archive that should be imported. Select the file using the Browse.... When the file is selected it will be loaded automatically. The file location is stored in the preferences and loaded as default value the next time the wizard is opened. In this case the user can use the Load button to simply load the default file. When the file is loaded advance the next wizard page.



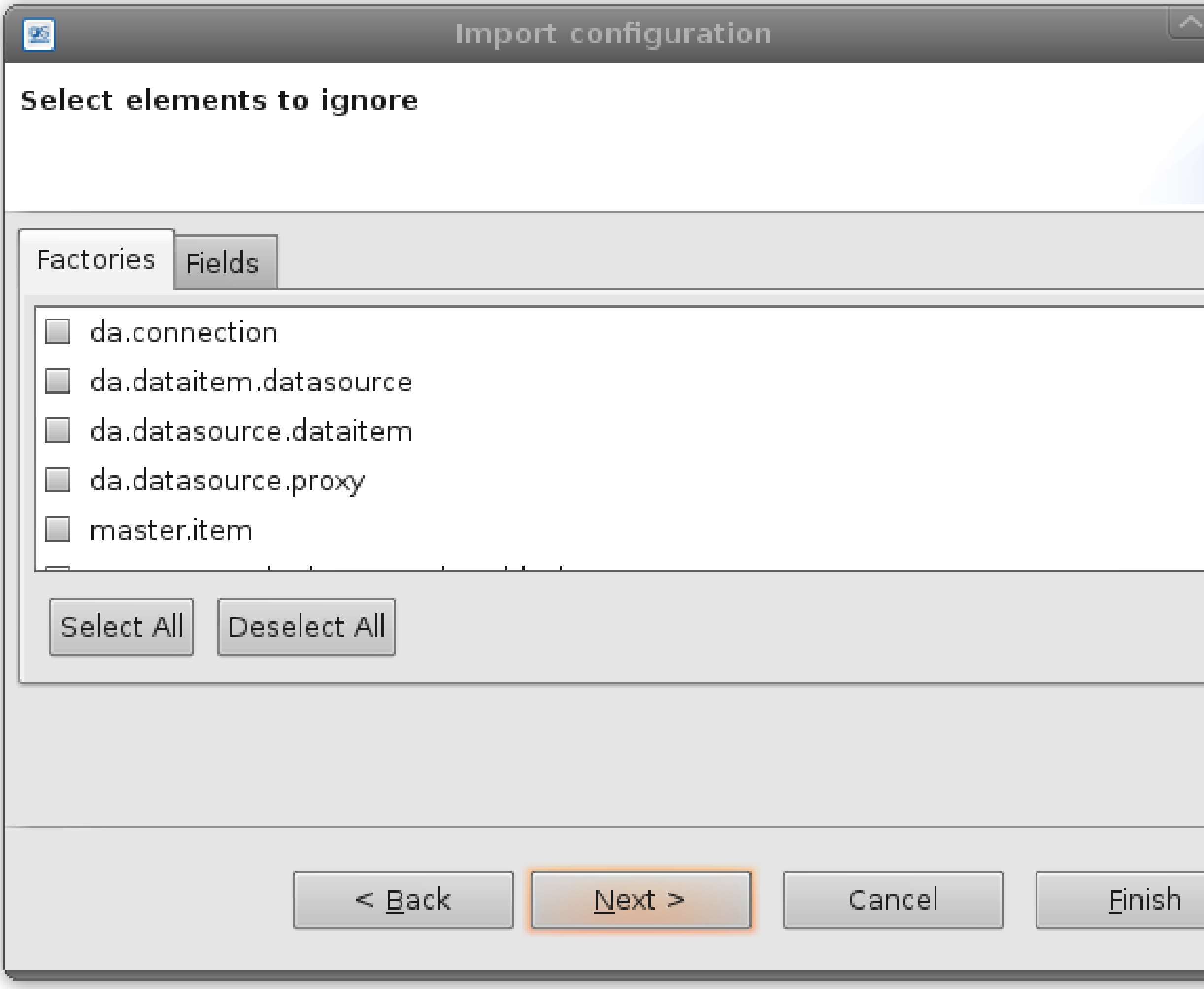
Select local data

The remote data used for the delta generation must be loaded. Normally this data is loaded from the CA of the server but for testing purposes it might be useful to make a delta using another local file. If you want to import data to the remote server use the button Load from server. After the data was loaded from the server advance to the next wizard page.



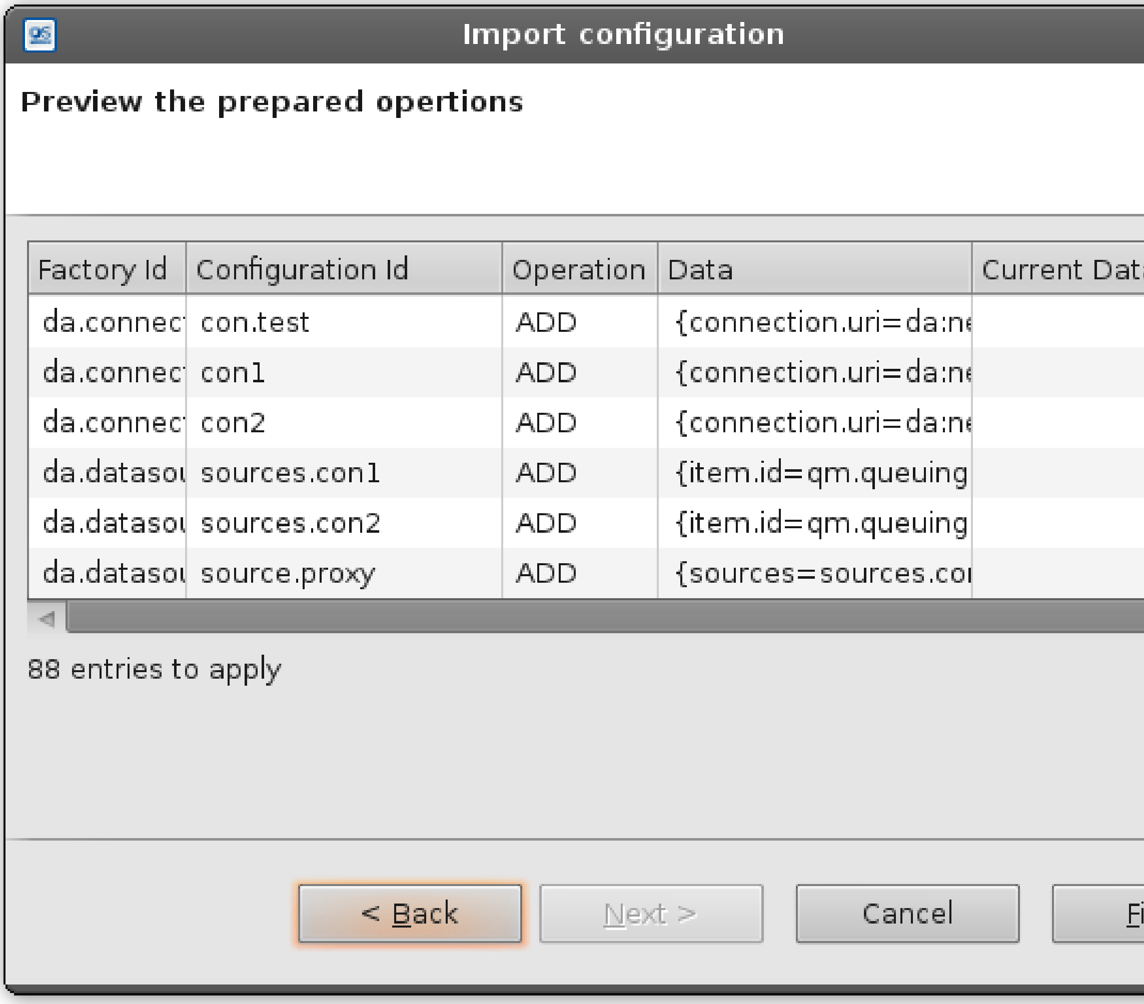
Select remote data

In this page the user has the option to select configuration elements that should be ignored when importing the new data. Factories that are selected will simply be ignore from the import process and neither created, updated nor deleted. Select factories that should be ignored by the import process and advance to the next wizard page.



Select elements to ignore

This last page of the wizard will show the delta and the operations that will be performed when the wizard is finished. This is the last chance to cancel the import process since nothing has been commited up to now. Pressing the Finish will start import process.



Review changes

7.2. OSGi™ based servers

This section focuses on the administration of OSGi™ based server installations.

7.2.1. Requirements & preconditions

The following requirements must be satisfied:

- Installed operating system
- Installed Java Version 5+
  - SUN or IBM JVM are recommended
  - A JRE is sufficient
  - The `JAVA_HOME` environment variable has the be set

7.2.2. Installation of administrative Tools

Note

The installation of the administrative tools is currently only supported on RPM based systems like RedHat Enterprise Linux™. Of course we want to expand this to other operating systems and distributions, but at the moment our focus is in RHEL.

It still is possible to install OpenSCADA™ on other system. It only is more work to do.

First copy the RPMs to the target system and install them using:

```
# rpm -Uvh *.rpm
```

7.2.3. Create a P2 base installation

It is now required to create a base server installation. This will install the OSGi™ container but will not install much functionality inside the container.

The location of the installation might be of interest to you or your system administrator. So it should be chosen carefully as is not recommended to be moved.

Create the P2 base installation by issuing the following command:

```
$ p2.create "targetDirectory"
```

This will create a new directory named `targetDirectory` and install the basic OSGi™ container. The command will fail if the directory already exists or the base path does not exist.

Next the custom artifacts that should be included in the setup need to be installed.

7.2.4. Deploying the OpenSCADA™ artifacts

The next step is to deploy the OpenSCADA™ bundles and features to the previously created P2 installation. Depending on the functionality you want to have in your server the selection of bundles or features might differ. If you just want “it all in” you can install the IU `org.openscada.deploy.feature.group`

```
$ p2.install test "org.openscada.deploy.feature.group"
```

It might also be that you need some project specific features that you need to install now.

7.2.5. Setting runtime parameters

In order to set up some Java™ system properties you can edit the file `laucher.properties` which contains a `key=value` scheme of system properties that will be set on startup of the container.

7.2.6. Starting the Equinox™ container

In order to start the container change the current directory to the installation directory issue the following command:

```
$ ./launcher
```

The OSGi™ container will start up and show the Equinox™ console. If you want to shut the application down you will need to issue the command “close”:

```
osgi> close
```

7.2.7. Customizing services

After installing the feature `org.openscada.deploy.feature.group` you will have all services and features installed. Bundles that you do not want to be in the container can be uninstalled by issuing the following command:

```
osgi> uninstall bundle.id
```

Although the rest of the bundles will be installed they are not started or active in any way. So they have to be started in order to provide the services you need. Of course at this point the OSGi™ dependency management kicks in an automatically activates bundles as you defined. This means that bundles might get started automatically if others depend on it or they might simply get “resolved” which means that they are in use but not activated (provide services). For more information about the OSGi™ lifecycle management see the OSGi™ documentation itself.

Equinox™ will remember which services were started and will restart them the next time the container is started.

7.2.8. A quick introduction into the Equinox™ OSGi™ console

List all installed bundles:

```
osgi> ss
```

List all installed bundles that match or partially match the string `openscada`:

```
osgi> ss openscada
```

Start a bundle named `org.openscada.ca.file`

```
osgi> start org.openscada.ca.file
```

Stop a bundle named `org.openscada.ca.file`

```
osgi> stop org.openscada.ca.file
```

Glossary

AE	Alarms and Events
CA	Configuration Administrator
DA	Data Access
Hive	A hive is a DA server instance.
HD	Historical Data
IU	An Installable Unit is a unit that can be installed in the P2 provisioning system. An installable unit will often have dependencies and metadata. It can be a simple OSGi™ bundle or a group of bundles, a feature or component.

Appendix A. GNU Free Documentation License

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties — for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See Copyleft [http://www.gnu.org/copyleft/].

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A “Massive Multiauthor Collaboration” (or “MMC”) contained in the site means any set of copyrightable works thus published on the MMC site.

“CC-BY-SA” means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

“Incorporate” means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is “eligible for relicensing” if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.