D6.3 OS4ES Middleware Implementation
# D6.3  OS4ES Middleware Implementation

## Document Information

<table>
<thead>
<tr>
<th>Program</th>
<th>FP7-ICT-2013-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym</td>
<td>OS4ES</td>
</tr>
<tr>
<td>Grant agreement number</td>
<td>619302</td>
</tr>
<tr>
<td>Number of the Deliverable</td>
<td>D6.3</td>
</tr>
<tr>
<td>WP/Task related</td>
<td>WP6 / T6.3</td>
</tr>
<tr>
<td>Type (distribution level)</td>
<td>PU</td>
</tr>
<tr>
<td>Start date of project</td>
<td>01/07/2014</td>
</tr>
<tr>
<td>Date of delivery</td>
<td>25/08/2016</td>
</tr>
<tr>
<td>Status and Version</td>
<td>Final Version</td>
</tr>
<tr>
<td>Number of pages</td>
<td>71 pages</td>
</tr>
<tr>
<td>Document Responsible</td>
<td>HYPERTECH</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Antonis Papanikolaou, Eleana Chatzoplaki, Panos Andriopoulos, Petros Balios – Hypertech</td>
</tr>
<tr>
<td></td>
<td>Markus Breuers - FGH</td>
</tr>
<tr>
<td></td>
<td>Wolfgang Renz - HUAS</td>
</tr>
<tr>
<td></td>
<td>Stjepan Sucic - Koncar</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Casper van den Broek - TNO</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Michiel van ben Berghe - Stedin</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Mikel Fernandez, Maitena Ilardia - Tecnalia</td>
</tr>
</tbody>
</table>
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author/Reviewer</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>24.04.2016</td>
<td>Hypertech</td>
<td>ToC generation</td>
</tr>
<tr>
<td>0.2</td>
<td>16.05.2016</td>
<td>Hypertech</td>
<td>ToC finalization</td>
</tr>
<tr>
<td>0.5</td>
<td>10.07.2016</td>
<td>Hypertech</td>
<td>Document draft generation</td>
</tr>
<tr>
<td>0.6</td>
<td>20.07.2016</td>
<td>Hypertech</td>
<td>Internal document review</td>
</tr>
<tr>
<td>0.9</td>
<td>23.08.2016</td>
<td>TNO</td>
<td>Document peer review</td>
</tr>
<tr>
<td>1.0</td>
<td>25.08.2016</td>
<td>Hypertech</td>
<td>Final editing and proof-reading</td>
</tr>
</tbody>
</table>
Executive Summary

The heart of the OS4ES System is the middleware responsible for enabling the smooth and robust information exchange and communication between the various components of the system and between the system, the involved actors and the interconnected DERs.

The OS4ES middleware has been developed in order to comply with the emerging standardization efforts of the IEC 61850 standards family. Hence it relies on XMPP as the application level communication protocol. The software implementation is based on Openfire, the most popular and feature-rich, open source implementation of a XMPP server. For the communication of the various system actors/users through the middleware, two different types of middleware clients were also developed. The first is based on the IEC 61850-8-2 standard draft which describes extends the existing IEC 61850 communication protocol to web services. The second is a simple XMPP client that is used to handle communications between the OAA and the registry which tend to involve large payloads.

This document, as an addendum to the software prototype which is the actual deliverable, describes the middleware implementation along with its configuration and deployment settings.

The integration of the central communication platform to the other software entities will be realized into the subsequent Task 6.4 “Integration of individual components into the OS4ES prototype”.
# Table of contents

Document Information.......................................................................................................................... 2
Revision History........................................................................................................................................ 3
Executive Summary................................................................................................................................. 4
Table of contents................................................................................................................................... 5
List of Tables........................................................................................................................................... 7
List of Figures......................................................................................................................................... 8
1 Introduction.......................................................................................................................................... 9
  1.1 Document scope and objectives ................................................................................................. 9
  1.2 OS4ES System and its Middleware Reference Architecture & Implementation Principles .......... 9
  1.3 Notations, abbreviations and acronyms ..................................................................................... 11
2 Openfire Server ................................................................................................................................ 13
  2.1 Server machine setup .................................................................................................................. 13
  2.2 Initial configuration steps .......................................................................................................... 14
  2.3 Database configuration .............................................................................................................. 17
  2.4 User/Actor Authentication ....................................................................................................... 20
3 Security settings ............................................................................................................................... 23
  3.1 TLS/SSL Certificates................................................................................................................ 23
    3.1.1 Background ......................................................................................................................... 23
    3.1.2 Sun JDK 1.5 security tools ............................................................................................... 28
4 Middleware IEC 61850-XMPP Clients ............................................................................................ 31
5 Conclusions........................................................................................................................................ 32
6 Annexes ............................................................................................................................................. 33
  6.1 OS4ES Openfire database schema............................................................................................. 33
  6.2 Openfire MySQL database creation script ............................................................................... 43
  6.3 Installation instructions for middleware clients ......................................................................... 49

Intro Dependencies.................................................................................................................................. 49
  Windows development machine setup ............................................................................................. 49
  Linux/Ubuntu operating systems ....................................................................................................... 50
6.4 Middleware client API descriptions ................................................................. 52

6.4.1 IEC 61850 client - Class Os4esIec61850Client .................................................... 52

Field Summary ........................................................................................................... 52
Constructor Summary ................................................................................................. 52
Method Summary ........................................................................................................... 52
Methods inherited from class java.lang.Object .......................................................... 56
Field Detail .................................................................................................................... 56
Constructor Detail ......................................................................................................... 56
Method Detail ................................................................................................................ 56

6.4.2 IEC 61850 server - Class Os4esIec61850Server ..................................................... 69

Constructor Summary ................................................................................................. 69
Method Summary ........................................................................................................... 69
Methods inherited from class java.lang.Object .......................................................... 69
Constructor Detail ......................................................................................................... 69
Method Detail ................................................................................................................ 70

References ..................................................................................................................... 71
List of Tables

Table 1 - Acronyms list .......................................................................................................................... 11
List of Figures

Figure 1 – OS4ES Middleware Reference Architecture................................................................. 10
Figure 2 – Blowfish encryption algorithm ......................................................................................... 14
Figure 3 – Openfire Server web browser setup Language selection step ........................................... 16
Figure 4 – Openfire Server web browser setup Server settings step ................................................. 16
Figure 5 – Openfire Server web browser setup Database settings step ............................................ 18
Figure 6 – Openfire web-browser setup database connection step .................................................... 19
Figure 7 – Openfire web-browser setup Profile settings step ............................................................ 21
Figure 8 – Openfire web-browser setup Profile Connection settings step ........................................... 22
Figure 9 – Openfire Client and Server Connection Security settings ............................................... 25
Figure 10 – Openfire web-browser administrative console SSL enabling configuration ............... 26
Figure 11 – Openfire Self-signed server certificates ......................................................................... 27
1 Introduction

1.1 Document scope and objectives

The purpose of this document is to describe the implementation of the OS4ES Semantic Middleware, the central communication service bus of the whole OS4ES platform. Towards the coverage of all the defined specifications given in a D6.1 “Semantic Middleware Architecture Specification”, various development/implementation options and configuration settings are analysed.

This document accompanies the middleware software prototype, which is the project deliverable, in order to acquaint the reader with the process of installing and configuring it. Descriptions of key functionalities are also provided.

In this document, the Middleware Reference Architecture and its design principles are shortly reminded in the introduction. In the second Chapter, the selection of the open-source Openfire XMPP Server is justified along with the analysis of its deployment and configuration. Extra taken security measurements are shown in Chapter 3 with their exact implementation steps. This deliverable also include 4 annexes with important information regarding the implementation and use of the middleware. They include the necessary database schema and the script to create it, the installation instructions for the software packages as well as the documentation of the API of the middleware client.

Please note that, as explained in the project report for the second period as well as deliverable 6.2.2, the consortium has opted to switch to continuous integration for the software development and integration approach. This has had an impact on the maturity and feature completeness of the OS4ES system components. Deliverable D6.4, the result of the integration task, will document the various integrated versions of the OS4ES reference implementation as well as their features in detail.

1.2 OS4ES System and its Middleware Reference Architecture & Implementation Principles

The Semantic Middleware of the OS4ES platform is the component responsible for performing the entire realizable message transferring between the separate components of the system and ensuring reliable and secure message delivery. Playing the role of the central communication layer, the Middleware Software component will sit between the different peripheral components of the platform and provide messaging services, such as concurrency, transaction management, threading and messaging, in order to facilitate all communication
processes. Message transactions between different entities will be based on IEC 61850 protocol and XML-based data models.

During the OS4ES Middleware implementation phase, the reference architecture as specified in Deliverable 6.1 slightly differentiated (Figure 1). The aforementioned difference focus on the additional capability the Middleware can offer for communication between the Registry and the OAA applications through XML over XMPP messages. In this final reference architecture, two kinds of communication are foreseen,

- IEC 61850 for communications to and from the all the connected DER systems
- XML over XMPP for communication between the aggregator application and the registry

![Figure 1 – OS4ES Middleware Reference Architecture](image-url)
Introduction

The conceptual architecture design was driven by the objective to allow the adaptation to any communication protocol, middleware and the various OS4ES components data models and communication standards.

The whole design was driven by the fact that in contradiction to the heterogeneity of the various components and actors of the system, their interoperability and mediate interaction should be ensured. The whole architecture is server based; any communication between the clients is done by exchanging messages over a central server farm.

Based on the analysis and outcomes of previous deliverables of the project, the following technology selections were made regarding the OS4ES platform components:

- OS4ES aims to be interoperable with any legacy DER system. The main components of the OS4ES middleware (the core, conversion layer and registry) will natively use an extension of the IEC 61850 data model. This provides natural support for DER systems compatible with the IEC 61850 protocol. In order to communicate with other DER systems as well, OS4ES will provide dedicated gateway components which will provide the necessary conversion functionality. For the reference implementation this conversion functionality will at least be able to handle Modbus-based DER systems that are available in the two lab test sites in Bilbao and Hamburg.

- XML over XMPP messaging will be also supported by the Middleware in order to address the communication requirements of the registry and the OAA that cannot be met by IEC 61850 due to the nature of the message characteristics. For more information please refer to deliverable 6.2.2 [1] of the project.

- The OAA applications will interact with the middleware using an Application Programming Interface (API) that is compatible with CIM (IEC 61970). This interface has been described in deliverable 5.3, which outlines the aggregator applications. The middleware client coupled to the OAA gateway ensure the smooth communication from the CIM-based OAA application to the IEC 61850-based DER systems.

- The selected communication protocol that will facilitate all the essential communication needs of the system components will be the Extensible Messaging and Presence Protocol (XMPP) [2]. More details regarding this choice are available in deliverable 6.1 [3].

1.3 Notations, abbreviations and acronyms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol is an application protocol for distributed, collaborative, hypermedia information systems</td>
</tr>
</tbody>
</table>
**XML**

eXtensible Markup Language is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable.

**XMPP**

Extensible Messaging and Presence Protocol is a communication protocol for message-oriented middleware based on XML (Extensible Markup Language)

**RTC**

Real Time Collaboration

**DES**

Data Encryption Standard

**VM**

Virtual Machine

**AES**

Advanced Encryption Standard

**LDAP**

Lightweight Directory Access Protocol

**OAA**

OS4ES Aggregator Application
2 Openfire Server

For the OS4ES XMPP Server Farm, the Openfire open-source framework [4] was chosen as the base of this component implementation integrated with the suitable configuration and extensions. Openfire is a real time collaboration (RTC) server licensed under the Open Source Apache 2.0 License [5]. It uses the widely adopted XMPP instant messaging open protocol, which is the one that was selected during the middleware specifications development, see deliverable 6.1 [3]. Openfire is easy to setup and administer, and at the same time offers improved security and performance compared to all other open-source implementations.

2.1 Server machine setup

For installing the Openfire server, a 64-bit Linux machine is used with Linux 3.8.0-34-generic x86_64 as the operating system version. The Openfire server version used available at Openfire Home Page is 4.0.2. During the server setup the following steps where followed.

Between the Red Hat package manager and the tar.gz build option, the second was chosen. The downloaded archive was extracted to /opt path by using the following commands:

`tar -xzvf openfire_4_0_2.tar.gz`
`mv openfire /opt`

The tar.gz build does not contain a bundled Java runtime (JRE), so JDK or JRE 1.7.0 or later should be installed. In the current deployment Java(TM) SE Runtime Environment (build 1.7.0_80-b15) on Java HotSpot(TM) 64-Bit Server VM (Virtual Machine) (build 24.80-b11, mixed mode) was already available on the hosting machine.

After the zipped file extraction, the available files in your distribution were as follows:

```
openfire/
|-- readme.html
|-- license.html
|-- conf/
|-- bin/
|-- jre/
|-- lib/
|-- plugins/
| |-- admin/
| |-- resources/
| | |-- database/
| | |-- security/
| | |-- documentation/
```

More specifically:

- The `conf` directory is where Openfire stores configuration files.
- The `bin` directory contains the server executables. Depending on the installed distribution, different executables are available.
- The **jre** directory contains a Java runtime that is bundled with the Windows and RPM versions of Openfire.
- The **lib** directory contains libraries necessary for running Openfire.
- The **plugins** directory contains server plugins. By default, Openfire ships with a web-based admin console plugin.
- The **resources/database** directory contains SQL schema files to create new Openfire databases, as well as upgrade scripts for existing installations.
- The **resources/security** directory is where Openfire maintains keystores to support SSL connection security.
- The **documentation** directory contains server documentation.

After the preceding installation process, the Openfire Server is able to start and stop using the `bin/openfire` script of the Openfire installation:

```
# ./openfire
Usage: ./openfire {start|stop}
# ./openfire start
Starting openfire
```

### 2.2 Initial configuration steps

The first time that the Openfire server starts, a set of configuration steps should be performed via the browser at the path `http://localhost:9090`. The initial steps of the Openfire Server web-browser setup include the language selection (Figure 3), and the Server settings (Figure 4). In the last step, the XMPP Server domain name is defined, along with the administration console ports. Default ports were chosen 9090 and 9091 for secure admin console. Two types of security-related server property encryption are supported Blowfish and AES.

Blowfish is a symmetric block cipher that takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. It was designed in 1993 as a fast, free alternative to existing encryption algorithms and since then is slowly gaining acceptance as a strong encryption algorithm (Figure 2).
Welcome to Setup

Welcome to Openfire Setup. This tool will lead you through the initial setup of the server. Before you continue, choose your preferred language.

Choose Language

- Czech (cs_CZ)
- Deutsch (de)
- English (en)
- Español (es)
- Français (fr)
- Nederlands (nl)
- Polski (pl_PL)
- Português Brasileiro (pt_BR)
- Русский (ru_RU)
- Slovenčina (sk)
- 中文 (简体) Simplified Chinese (zh_CN)

Continue
Figure 3 – Openfire Server web browser setup Language selection step

Figure 4 – Openfire Server web browser setup Server settings step
AES (acronym of Advanced Encryption Standard) is a symmetric encryption algorithm. The algorithm was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. Based on an available research literature [6], results have shown that Blowfish has a better performance than the other common encryption algorithms used. Since Blowfish has no known security weak points so far, it is an excellent candidate to be considered as a standard encryption algorithm and it was chosen for the XMPP server instance of the OS4ES reference implementation. On the contrary, AES showed poor performance results compared to other algorithms since it requires significant more processing power.

2.3 Database configuration

One of the most important steps is the XMPP Server database configuration (Figure 5). Connection with an external standard database was selected with the following configuration parameters.

- Database Driver Presets: MySql
- JDBC Driver Class: com.mysql.jdbc.Driver
- Hostname: 127.0.0.1 (localhost)
- Database name: os4es_openfire

Prior to this configuration step the os4es_openfire database is created at the MySQL server already existing at the host machine. You can find the database schema in Annex 6.1, along with the MySQL creation script in Annex 6.2.
Figure 5 – Openfire Server web browser setup Database settings step
Figure 6 – Openfire web-browser setup database connection step
2.4 User/Actor Authentication

The current version of the middleware uses two-factor authentication based on username and password. Accounts and credentials are handled centrally, they are generated by the Openfire administrator and distributed in a safe manner to the actor who wish to connect with the middleware. This scheme will be used for the initial system integration tests and the lab tests, where the number of middleware users is relatively small. Hence the current configuration of Openfire can be seen below. All credentials are stored encrypted in the Openfire database as explained in the previous section.

However, Openfire can be configured to authenticate users through an OpenLDAP server which is a much more scalable and easy to administer solution. This authentication method will be used in subsequent versions of the OS4ES middleware to improve security and scalability of the entire OS4ES system.

OpenLDAP is a free, open source implementation of the Lightweight Directory Access Protocol (LDAP) developed by the OpenLDAP Project. It is released under its own BSD-style license called the OpenLDAP Public License. This allows to use a single user and password source, rather than needing to maintain two or more separate databases. This is highly recommended towards the enhancement of the connectivity security provided by the XMPP Server. To begin, on the Profile Settings screen, the Directory Server (LDAP) option was selected (Figure 7). The OpenLDAP server connection settings were filled as shown in Figure 8 with the primary domain name and the top-level domain. Subsequent to the successful configuration of the profile settings, the whole Openfire Server setup is concluded and the XMPP server is ready for XMPP clients to connect.
Figure 7 – Openfire web-browser setup Profile settings step
Figure 8 – Openfire web-browser setup Profile Connection settings step
3 Security settings

As explained in the previous chapter, the first integrated version of the OS4ES reference implementation supports only user authentication using a standard two-factor authentication method. However, one of the most critical requirements to be fulfilled during the Middleware implementation is to assure secure message exchange. In the next version of the OS4ES Middleware, message encryption during communication will be also be supported utilizing the TLS/SSL Certificates as explained below. This feature is currently lacking in the first integrated OS4ES reference implementation to be delivered in September 2016. The middleware can already support it, but the necessary support on the client side (e.g. key management) is not ready yet, hence it will be implemented in the following version of the integrated system. The delivery roadmap and versions of the OS4ES reference implementations will be provided in deliverable 6.4 in Month 28 of the project.

3.1 TLS/SSL Certificates

Openfire's SSL support is built using the standard Java security SSL implementation (javax.net.ssl.SSLServerSocket). You can see SSL enabling parameter on the web-browser administrative console in Figure 10. Moreover, it can generate self-signed DSA and RSA certificates through its Administrator Web Console (Server Settings | Server Certificates) (Figure 11). These can be used for server-to-server as well as for client-server communication (Server Settings | Security Settings) as shown in Figure 9.

3.1.1 Background

A server SSL connection uses two sets of certificates to secure the connection. The first set is called a "keystore". The keystore contains the keys and certificates for the server. These security credentials are used to prove to clients that the server is legitimately operating on behalf of a particular domain. If your server will only need to act as one domain, you only need one key entry and certificate in the keystore. Keys are stored in the keystore under aliases. Each alias corresponds to a domain name (e.g. "example.com").

The second set of certificates is called the "truststore" and is used to verify that a client is legitimately operating on behalf of a particular user. In the vast majority of cases, the truststore is empty and the server will not attempt to validate client connections using SSL. Instead, the XMPP authentication process verifies users in-band. However, it may be wished to require SSL authentication for certain clients when security is especially important and the number of client connections to the server is relatively small.
The number of the concurrent connections that an Openfire Server can support depends on the hosting operating system and infrastructure as well as on the selected messaging security settings. Assuming a relatively “large” cloud server (i.e. two Intel Xeon E5-2673 v3 at 2.4 GHz cores and 16GB RAM & no bandwidth bottlenecks) and that TLS/SSL certificates are activated, the maximum number of the concurrent connections handled is about 25,000. In case there is need for extending this capacity two solutions are available: usage of more XMPP Servers through federation or usage of a separate component called a multiplexer. Multiplexers are scalability components that consolidate messenger connections into a single TCP connection between physical servers [7]. To support large deployments, with hundreds of thousands of concurrent connections, connection multiplexers are used to improve server scalability. This component opens only one single connection to the XMPP server. In addition to scalability, you can install the multiplexer outside the firewall while leaving the server inside the firewall to protect it from unauthorized external access.

In the next version of the middleware TLS will be used to secure all connections of between the OS4ES components. The use of additional XMPP server or multiplexers will be evaluated and decided on an ad-hoc basis in each case. The deployment of more cloud server machines and XMPP servers is straightforward and can be easily done if necessary.

Certificates attempt to guarantee that a particular party is who they claim to be. Certificates are trusted based on who signed the certificate. If only light security is required, e.g. systems are deploying for internal use on trusted networks, "self-signed" certificates can be utilized. Self-signed certificates encrypt the communication channel between client and server. However, the client must verify the legitimacy of the self-signed certificate through some other channel. The most common client reaction to a self-signed certificate is to ask the user whether to trust the certificate, or to silently trust the certificate is legitimate.
Figure 9 – Openfire Client and Server Connection Security settings
Figure 10 – Openfire web-browser administrative console SSL enabling configuration
Security settings

Figure 11 – Openfire Self-signed server certificates
The advantage of a self-signed certificate is that they can be created for free which is great when cost is a major concern, or for testing and evaluation. In addition, a self-signed certificate can be safely used if the certificate in use is legitimate. So if a system administrator creates a self-signed certificate, then personally installs it on a client's truststore (so that the certificate is trusted) you can be assured that the SSL connection will only work between the client and the correct server.

Within the project lifetime, the possibility of using well-known certificate authorities in order to generated trusted, signed certificates will also be explored in order to pave the way for an even more secure version of the OS4ES system that can be deployed in real-life contexts.

3.1.2 Sun JDK 1.5 security tools

The Sun JDK (version 1.5.x) ships with all the security tools you need to configure TLS/SSL with Openfire. The most important is the keytool located in the JAVA_HOME/bin directory of the JDK. Sun JVMs persist keystores and truststores on the filesystem as encrypted files. The keytool is used to create, read, update, and delete entries in these files. Openfire ships with a self-signed "dummy" certificate designed for initial evaluation testing. You will need to adjust the default configuration for most deployments.

In order to configure SSL on your server you need complete the following tasks:

1. Decide on your Openfire server's domain.
2. Create a self-signed SSL server certificate for your server domain. Note: you may already have one if your Openfire server domain matches an existing web domain with SSL. If so, you can skip to step 4.
3. [Optional] Have a certificate authority (CA) certify the SSL server certificate.  
   a. Generate a certificate signing request (CSR).
   b. Submit your CSR to a CA for signing.
4. Import the server certificate into the keystore. Note: if you are going to use a self-signed certificate generated in step 2, the certificate is already imported and you can skip this step.
5. Remove default certificates from the keystore.
6. Import client certificates into the truststore.
7. Adjust the Openfire configuration with proper keystore and truststore settings.

1. Decide on a Server Domain

The Openfire server domain should match the host name of the server; for example, "example.com". Your user accounts will have addresses with the format "user@example.com" like email addresses. We'll assume the domain is "example.com" for the rest of the examples.
2. Create a self-signed server certificate

In order to create a self-signed server certificate go to the command line and change directories to the `resources/security` directory of your Openfire installation. You should see the default `keystore` and `truststore` files. First, you should change the default keystore password:

```
keytool -storepassw -keystore keystore
```

Keytool will ask for the old password (by default it is `changeit`) then the new password. Now we'll create a certificate using the keytool:

```
keytool -genkey -keystore keystore -alias example.com
```

where you should substitute your server's name for `example.com`. The keytool will ask for the store password, then several pieces of information required for the certificate. Enter all the information but remember to complete with your server's name when asked for your first and last name. After you have entered all the required information, keytool will ask you to verify the information and set a key password. You must use the same key password as the store password. By default you get this by simply hitting 'enter' when prompted for a key password.

If you later change the keystore password remember to change the entries' password as well using the keytool:

```
keytool -keypassword -alias example.com -keystore keystore
```

Keytool will create certificates using the DSA algorithm by default. Some clients expect the server to have RSA certificates or they will fail to use TLS/SSL. Therefore, it is a good idea to also create RSA certificates in your keystore. To create certificates with the RSA algorithm you need to specify the algorithm to use like this:

```
keytool -genkey -keystore keystore -alias example.com -keyalg RSA
```

3. Obtain a CA signed certificate

If you decide to get a CA signed certificate, you must first export the certificate in the standard CSR format. You can do this with the keytool:

```
keytool -certreq -keystore keystore -alias example.com -file certificate_file
```

Where you should substitute your server's name for `example.com` and the name of the certificate file you wish to produce for `certificate_file`. Submit the generated CSR to the CA and follow their instructions to get it signed.

4. Import server certificates

If you had a CA sign your server certificate, or if you have an existing SSL certificate, you must import it using the keytool.

```
keytool -import -keystore keystore -alias example.com -file signed_certificate_file
```
It is important that the alias not already have an associated key or you'll receive an error.

5. Remove default certificates

After importing the certificate, the default certificates should be removed using the keytool.

```bash
keytool -delete -keystore keystore -alias rsa
keytool -delete -keystore keystore -alias dsa
```

6. Import client certificates

If you require clients to verify themselves using certificates, obtain their certificates and import them into the truststore file rather than the keystore. First, you should change the default truststore password:

```bash
keytool -storepasswd -keystore truststore
```

keytool will ask for the old password (by default it is `changeit`) then the new password. Now import each certificate using the keytool:

```bash
keytool -import -keystore truststore -alias user_name -file certificate_file
```

7. Configure Openfire

Open the Openfire Admin Console in your favorite browser and add or change the following system properties:

- `xmpp.socket.ssl.active` -- set to 'true' to active SSL
- `xmpp.socket.ssl.port` -- the port to use for SSL (default is 5223 for XMPP)
- `xmpp.socket.ssl.storeType` -- the store type used ("JKS" is the Sun Java Keystore format used by the JDK keytool). If this property is not defined, Openfire will assume a value of "jks".
- `xmpp.socket.ssl.keystore` -- the location of the keystore file relative to your Openfire installation root directory. You can leave this property blank to use the default keystore.
- `xmpp.socket.ssl.keypass` -- the keystore/key password you changed in step 2.
- `xmpp.socket.ssl.truststore` -- leave blank to not use a truststore, otherwise the location of the truststore file relative to your Openfire installation root directory.
- `xmpp.socket.ssl.trustpass` -- the truststore/key password you changed in step 6.
4 Middleware IEC 61850-XMPP Clients

Dedicated OS4ES clients were implemented in Java to support both:

- XMPP communication based on the IEC 61850 protocol as described in the standard drat IEC 61850-8-2 using the 61850 protocol stack implementation delivered as part of WP3, and,
- XML-based messaging using plain XMPP.

The OS4ES clients supporting IEC 61850 communication protocol were developed within the scope of the middleware component.

The source code of the middleware clients is uploaded on the GitLab repository¹, and the required installation instructions are also provided in Annex 6.3. Documentation of this source code is provided in Annex 6.4 in the present document in the form of their API outlining the functionality of the two most complex middleware client component:

- the IEC 61850 client, which is required by the Registry and OAAplication in order to communicate with the DER system
- the IEC 61850 server, which is required at the DER system controller side to facilitate communications with all other OS4ES components.

The final middleware client component is the one that facilitates communications between the registry and the OAAplication, which only implements a vanilla XMPP client as shown in Figure 1. The open-source Smack library has been used to implement this. Its example documentation can be found at the Smack API javadoc [8].
5 Conclusions

The implementation of the OS4ES Middleware as well as instructions for its installation were presented in this document. The Openfire XMPP Server as the backbone of the OS4ES middleware was chosen due its maturity and acceptance from the IT community. Additional plugins and extension were made on the core distribution of the solution adjusting to the project needs. Extra security measures were taken towards the security of the personal information of the pilot site users involved. Subsequent to Task 6.3, testing and integration of the component will give feedback for small refinements and improvements.
## 6 Annexes

### 6.1 OS4ES Openfire database schema

#### ofGroup (user Group data)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>VARCHAR</td>
<td>50</td>
<td>Group Name (Primary Key)</td>
</tr>
<tr>
<td>description</td>
<td>VARCHAR</td>
<td>255</td>
<td>Group Description</td>
</tr>
</tbody>
</table>

#### ofGroupProp (name-value associations for a Group)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>VARCHAR</td>
<td>50</td>
<td>Group Name (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Group Property Name (Primary Key)</td>
</tr>
<tr>
<td>propValue</td>
<td>VARCHAR</td>
<td>4000</td>
<td>Group Property Value</td>
</tr>
</tbody>
</table>

#### ofGroupUser (associates Users with Groups)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>groupName</td>
<td>VARCHAR</td>
<td>50</td>
<td>Group Name (Primary Key)</td>
</tr>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>100</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>administrator</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Administrator (Boolean) (Primary Key)</td>
</tr>
</tbody>
</table>

#### ofID (used for unique ID sequence generation)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>idType</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID type (e.g., Group, User, Roster) (Primary Key)</td>
</tr>
<tr>
<td>id</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Next available block of ID’s (Used for Database-Independent ID Sequence Generator)</td>
</tr>
</tbody>
</table>

#### ofOffline (offline message storage)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>messageID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of stored message (Primary Key)</td>
</tr>
<tr>
<td>creationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Date message stored</td>
</tr>
<tr>
<td>messageSize</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Size of message in bytes</td>
</tr>
<tr>
<td>stanza</td>
<td>TEXT</td>
<td>n/a</td>
<td>The message text</td>
</tr>
</tbody>
</table>
### ofPresence (offline presence)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>64</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>offlinePresence</td>
<td>TEXT</td>
<td>n/a</td>
<td>Presence message set as user logged off</td>
</tr>
<tr>
<td>offlineDate</td>
<td>CHAR</td>
<td>15</td>
<td>Date message stored</td>
</tr>
</tbody>
</table>

### ofPrivate (Private data storage)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Name of the private entry (Primary Key)</td>
</tr>
<tr>
<td>namespace</td>
<td>VARCHAR</td>
<td>200</td>
<td>Namespace of the private entry (Primary Key)</td>
</tr>
<tr>
<td>privateData</td>
<td>TEXT</td>
<td>n/a</td>
<td>Value of the private data</td>
</tr>
</tbody>
</table>

### ofUser (User data)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>plainPassword</td>
<td>VARCHAR</td>
<td>32</td>
<td>Plain-text password data</td>
</tr>
<tr>
<td>encryptedPassword</td>
<td>VARCHAR</td>
<td>255</td>
<td>Encrypted password data (default)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Name</td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR</td>
<td>100</td>
<td>Email Address</td>
</tr>
<tr>
<td>creationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Creation Date</td>
</tr>
<tr>
<td>modificationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Last Modified Date</td>
</tr>
</tbody>
</table>

### ofUserProp (name-value associations for a User)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>User Property Name (Primary Key)</td>
</tr>
<tr>
<td>propValue</td>
<td>VARCHAR</td>
<td>4000</td>
<td>User Property Value</td>
</tr>
</tbody>
</table>

### ofUserFlag (special flags set on a User (like disabled))

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>64</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>User Property Name (Primary Key)</td>
</tr>
<tr>
<td>startTime</td>
<td>CHAR</td>
<td>15</td>
<td>Time when the flag is to start being effective (null for ‘now’)</td>
</tr>
</tbody>
</table>
### endTime

**Type:** CHAR  
**Length:** 15  
**Description:** Time when the flag is to end being effective (null for 'forever')

### ofRoster (buddy rosters or lists)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of roster (Primary Key)</td>
</tr>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name</td>
</tr>
<tr>
<td>jid</td>
<td>TEXT</td>
<td>n/a</td>
<td>The address of the roster entry</td>
</tr>
<tr>
<td>sub</td>
<td>NUMBER</td>
<td>n/a</td>
<td>The subscription status of the entry</td>
</tr>
<tr>
<td>ask</td>
<td>NUMBER</td>
<td>n/a</td>
<td>The ask status of the entry</td>
</tr>
<tr>
<td>recv</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating the entry is a roster request that was received</td>
</tr>
<tr>
<td>nick</td>
<td>VARCHAR</td>
<td>255</td>
<td>The nickname assigned to this roster entry</td>
</tr>
</tbody>
</table>

### ofRosterGroups (Groups of buddy entries in a roster)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rosterID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Roster ID (Primary Key)</td>
</tr>
<tr>
<td>rank</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Position of the entry (Primary Key)</td>
</tr>
<tr>
<td>groupName</td>
<td>VARCHAR</td>
<td>255</td>
<td>The user defined name for this roster group</td>
</tr>
</tbody>
</table>

### ofPrivacyList (Users privacy lists)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Name of the privacy list (Primary Key)</td>
</tr>
<tr>
<td>isDefault</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating if this is the default privacy list of the user</td>
</tr>
<tr>
<td>list</td>
<td>TEXT</td>
<td>n/a</td>
<td>XML representation of the privacy list</td>
</tr>
</tbody>
</table>

### ofVCard (vCard contact information)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>32</td>
<td>User Name (Primary Key)</td>
</tr>
<tr>
<td>vcard</td>
<td>TEXT</td>
<td>n/a</td>
<td>Value of the vCard entry</td>
</tr>
</tbody>
</table>

### ofVersion (contains product version information)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
</table>
### ofProperty (server properties)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>50</td>
<td>Name of the item that version information is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>being tracked for (Primary Key)</td>
</tr>
<tr>
<td>version</td>
<td>INTEGER</td>
<td>n/a</td>
<td>The version number</td>
</tr>
</tbody>
</table>

### ofExtComponentConf (external components configuration)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subdomain</td>
<td>VARCHAR</td>
<td>255</td>
<td>Subdomain of the external component (Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Key)</td>
</tr>
<tr>
<td>secret</td>
<td>VARCHAR</td>
<td>255</td>
<td>Shared secret key of the external component</td>
</tr>
<tr>
<td>permission</td>
<td>VARCHAR</td>
<td>10</td>
<td>Permission that indicates if the component is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>allowed to connect to the server</td>
</tr>
</tbody>
</table>

### ofRemoteServerConf (remote servers' configuration)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmppDomain</td>
<td>VARCHAR</td>
<td>255</td>
<td>Domain of the external component (Primary Key)</td>
</tr>
<tr>
<td>remotePort</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Port of the remote server to connect to</td>
</tr>
<tr>
<td>permission</td>
<td>VARCHAR</td>
<td>10</td>
<td>Permission that indicates if the remote server</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is allowed to connect to the server</td>
</tr>
</tbody>
</table>

### ofSecurityAuditLog (logging of security events)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of audit message (Primary Key)</td>
</tr>
<tr>
<td>username</td>
<td>VARCHAR</td>
<td>64</td>
<td>user who performed the action</td>
</tr>
<tr>
<td>entryStamp</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Timestamp when event occurred</td>
</tr>
<tr>
<td>summary</td>
<td>VARCHAR</td>
<td>255</td>
<td>Summary of what occurred in event</td>
</tr>
<tr>
<td>node</td>
<td>VARCHAR</td>
<td>255</td>
<td>Node where event occurred</td>
</tr>
<tr>
<td>details</td>
<td>TEXT</td>
<td>n/a</td>
<td>Verbose details of what occurred</td>
</tr>
</tbody>
</table>

### ofMucService (A Groupchat service)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Name</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------</td>
<td>--------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>serviceID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of service (Indexed)</td>
</tr>
<tr>
<td>subdomain</td>
<td>VARCHAR</td>
<td>255</td>
<td>Subdomain of service (Primary Key)</td>
</tr>
<tr>
<td>description</td>
<td>VARCHAR</td>
<td>255</td>
<td>Description of service</td>
</tr>
<tr>
<td>isHidden</td>
<td>NUMBER</td>
<td>n/a</td>
<td>1 if hidden from admin interface lists, 0 of normal</td>
</tr>
</tbody>
</table>

**ofMucServiceProp (name-value associations for a Groupchat service)**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of service (Primary Key)</td>
</tr>
<tr>
<td>Name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Property Name (Primary Key)</td>
</tr>
<tr>
<td>propValue</td>
<td>TEXT</td>
<td>n/a</td>
<td>Property Value</td>
</tr>
</tbody>
</table>

**ofMucRoom (Groupchat room data)**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roomID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of room (Primary Key)</td>
</tr>
<tr>
<td>creationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Creation Date</td>
</tr>
<tr>
<td>modificationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Last Modified Date</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>50</td>
<td>Name of the room used as the public ID</td>
</tr>
<tr>
<td>naturalName</td>
<td>VARCHAR</td>
<td>255</td>
<td>Natural name of the room</td>
</tr>
<tr>
<td>description</td>
<td>VARCHAR</td>
<td>255</td>
<td>Room Description</td>
</tr>
<tr>
<td>canChangeSubject</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether participants can change the subject</td>
</tr>
<tr>
<td>maxUsers</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max number of room occupants</td>
</tr>
<tr>
<td>canChangeSubject</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether participants can change the subject or not</td>
</tr>
<tr>
<td>publicRoom</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the room will be listed in the directory or not</td>
</tr>
<tr>
<td>moderated</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the room is moderated or not</td>
</tr>
<tr>
<td>membersOnly</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the room is members-only or not</td>
</tr>
<tr>
<td>canInvite</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether occupants can invite other users</td>
</tr>
<tr>
<td>roomPassword</td>
<td>VARCHAR</td>
<td>50</td>
<td>Password Data for joining the room</td>
</tr>
<tr>
<td>canDiscoverJID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether real JID of occupants is public or not</td>
</tr>
</tbody>
</table>
### ofMucRoomProp (name-value associations for a Groupchat room)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roomID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of room (Primary Key)</td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR</td>
<td>100</td>
<td>Property Name (Primary Key)</td>
</tr>
<tr>
<td>propValue</td>
<td>VARCHAR</td>
<td>4000</td>
<td>Property Value</td>
</tr>
</tbody>
</table>

### ofMucAffiliation (affiliation of room users)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roomID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of room (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>TEXT</td>
<td>n/a</td>
<td>User JID (Primary Key)</td>
</tr>
<tr>
<td>affiliation</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Number representing the affiliation level</td>
</tr>
</tbody>
</table>

### ofMucMember (rooms members information)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roomID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of room (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>TEXT</td>
<td>n/a</td>
<td>User JID (Primary Key)</td>
</tr>
<tr>
<td>nickname</td>
<td>VARCHAR</td>
<td>255</td>
<td>Reserved nickname of the member</td>
</tr>
</tbody>
</table>

### ofMucConversationLog (rooms conversations log)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>roomID</td>
<td>NUMBER</td>
<td>n/a</td>
<td>ID of room</td>
</tr>
<tr>
<td>sender</td>
<td>TEXT</td>
<td>n/a</td>
<td>JID of the user that sent the message to the room</td>
</tr>
<tr>
<td>nickname</td>
<td>VARCHAR</td>
<td>255</td>
<td>Nickname used by the user when sending the message</td>
</tr>
<tr>
<td>Column Name</td>
<td>Type</td>
<td>Length</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>--------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>logTime</td>
<td>VARCHAR</td>
<td>15</td>
<td>Date when the message was sent to the room</td>
</tr>
<tr>
<td>subject</td>
<td>VARCHAR</td>
<td>50</td>
<td>New subject changed with the message</td>
</tr>
<tr>
<td>body</td>
<td>TEXT</td>
<td>n/a</td>
<td>Body of the message</td>
</tr>
</tbody>
</table>

### ofPubsubNode (nodes of the pubsub service)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node (Primary Key)</td>
</tr>
<tr>
<td>leaf</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the node is a leaf or collection node</td>
</tr>
<tr>
<td>creationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Creation Date</td>
</tr>
<tr>
<td>modificationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Last Modified Date</td>
</tr>
<tr>
<td>parent</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the parent node (if any)</td>
</tr>
<tr>
<td>deliverPayloads</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether payloads are included in notifications</td>
</tr>
<tr>
<td>maxPayloadSize</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max size of the payload in bytes</td>
</tr>
<tr>
<td>persistItems</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the node will persist published items</td>
</tr>
<tr>
<td>maxItems</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max number of items to persist</td>
</tr>
<tr>
<td>notifyConfigChanges</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when the node configuration has changed</td>
</tr>
<tr>
<td>notifyDelete</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when the node is deleted</td>
</tr>
<tr>
<td>notifyRetract</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when published items are deleted</td>
</tr>
<tr>
<td>presenceBased</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications to only users only</td>
</tr>
<tr>
<td>sendItemSubscribe</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send last published item to new subscribers</td>
</tr>
<tr>
<td>publisherModel</td>
<td>VARCHAR</td>
<td>15</td>
<td>Publisher model used by the node</td>
</tr>
<tr>
<td>subscriptionEnabled</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether subscriptions are allowed</td>
</tr>
<tr>
<td>configSubscription</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether new subscriptions must be configured to become active</td>
</tr>
<tr>
<td>accessModel</td>
<td>VARCHAR</td>
<td>10</td>
<td>Access model used by the node</td>
</tr>
<tr>
<td>payloadType</td>
<td>VARCHAR 100</td>
<td>Type of payload data to be provided at the node</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>bodyXSLT</td>
<td>VARCHAR 100</td>
<td>URL of an XSLT for transforming the payload format into a message body</td>
<td></td>
</tr>
<tr>
<td>dataformXSLT</td>
<td>VARCHAR 100</td>
<td>URL of an XSLT for transforming the payload format into Data Forms result</td>
<td></td>
</tr>
<tr>
<td>creator</td>
<td>VARCHAR 1024</td>
<td>JID of the entity that created the node</td>
<td></td>
</tr>
<tr>
<td>description</td>
<td>VARCHAR 255</td>
<td>Description of the node</td>
<td></td>
</tr>
<tr>
<td>language</td>
<td>VARCHAR 255</td>
<td>Default language of the node</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>VARCHAR 50</td>
<td>Name of the node</td>
<td></td>
</tr>
<tr>
<td>replyPolicy</td>
<td>VARCHAR 15</td>
<td>Policy that defines whether owners or publisher should receive replies to items</td>
<td></td>
</tr>
<tr>
<td>associationPolicy</td>
<td>VARCHAR 15</td>
<td>Policy that defines who may associate leaf nodes with a collection</td>
<td></td>
</tr>
<tr>
<td>maxLeafNodes</td>
<td>NUMBER n/a</td>
<td>Max number of leaf nodes that a collection node might have</td>
<td></td>
</tr>
</tbody>
</table>

### ofPubsubNodeJIDs (JIDs associated with nodes)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>VARCHAR</td>
<td>1024</td>
<td>JID of the entity (Primary Key)</td>
</tr>
<tr>
<td>associationType</td>
<td>VARCHAR</td>
<td>20</td>
<td>Type of association with the node</td>
</tr>
</tbody>
</table>

### ofPubsubNodeGroups (Roster groups associated with nodes)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node</td>
</tr>
<tr>
<td>rosterGroup</td>
<td>VARCHAR</td>
<td>100</td>
<td>Roster group of the node owner allowed to subscribe and retrieve items</td>
</tr>
</tbody>
</table>

### ofPubsubAffiliation (node affiliates)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>VARCHAR</td>
<td>1024</td>
<td>JID of the affiliate (Primary Key)</td>
</tr>
</tbody>
</table>
### ofPubsubItem (items published to nodes)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node (Primary Key)</td>
</tr>
<tr>
<td>id</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the published item (unique per node) (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>VARCHAR</td>
<td>1024</td>
<td>JID of the publisher</td>
</tr>
<tr>
<td>creationDate</td>
<td>VARCHAR</td>
<td>15</td>
<td>Creation Date</td>
</tr>
<tr>
<td>payload</td>
<td>TEXT</td>
<td>n/a</td>
<td>XML of the payload included in the published item</td>
</tr>
</tbody>
</table>

### ofPubsubSubscription (subscriptions to nodes)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>nodeID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the node (Primary Key)</td>
</tr>
<tr>
<td>id</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of the subscription (Primary Key)</td>
</tr>
<tr>
<td>jid</td>
<td>VARCHAR</td>
<td>1024</td>
<td>Address to receive notifications</td>
</tr>
<tr>
<td>owner</td>
<td>VARCHAR</td>
<td>1024</td>
<td>JID of the affiliate that owns the subscription</td>
</tr>
<tr>
<td>state</td>
<td>VARCHAR</td>
<td>15</td>
<td>State of the subscription (in the workflow)</td>
</tr>
<tr>
<td>deliver</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether notifications are enabled or not</td>
</tr>
<tr>
<td>digest</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether an entity wants to receive digests of notifications</td>
</tr>
<tr>
<td>digest_frequency</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Minimum number of milliseconds between sending any two notification digests</td>
</tr>
<tr>
<td>expire</td>
<td>VARCHAR</td>
<td>15</td>
<td>Date at which a leased subscription will end or has ended</td>
</tr>
<tr>
<td>includeBody</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether an entity wants to receive a message body in addition to the payload format</td>
</tr>
<tr>
<td>showValues</td>
<td>VARCHAR</td>
<td>30</td>
<td>Presence states for which an entity wants to receive notifications</td>
</tr>
<tr>
<td>subscriptionType</td>
<td>VARCHAR</td>
<td>10</td>
<td>Whether subscriber is subscribed to items or nodes (collection nodes only)</td>
</tr>
<tr>
<td>subscriptionDepth</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Receive notification from children up to certain depth (collection nodes only)</td>
</tr>
<tr>
<td>keyword</td>
<td>VARCHAR 200</td>
<td>Keyword that the event needs to match</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>---------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>

### ofPubsubDefaultConf (default configuration of nodes)

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Type</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>serviceID</td>
<td>VARCHAR</td>
<td>100</td>
<td>ID of service hosting the node (Primary Key)</td>
</tr>
<tr>
<td>leaf</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether configuration belongs to a leaf or collection node (Primary Key)</td>
</tr>
<tr>
<td>deliverPayloads</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether payloads are included in notifications</td>
</tr>
<tr>
<td>maxPayloadSize</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max size of the payload in bytes</td>
</tr>
<tr>
<td>persistItems</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether the node will persist published items</td>
</tr>
<tr>
<td>maxItems</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max number of items to persist</td>
</tr>
<tr>
<td>notifyConfigChanges</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when the node configuration has changed</td>
</tr>
<tr>
<td>notifyDelete</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when the node is deleted</td>
</tr>
<tr>
<td>notifyRetract</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications when published items are deleted</td>
</tr>
<tr>
<td>presenceBased</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send notifications to only users only</td>
</tr>
<tr>
<td>sendItemSubscribe</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether to send last published item to new subscribers</td>
</tr>
<tr>
<td>publisherModel</td>
<td>VARCHAR</td>
<td>15</td>
<td>Publisher model used by the node</td>
</tr>
<tr>
<td>subscriptionEnabled</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Flag indicating whether subscriptions are allowed</td>
</tr>
<tr>
<td>accessModel</td>
<td>VARCHAR</td>
<td>10</td>
<td>Access model used by the node</td>
</tr>
<tr>
<td>language</td>
<td>VARCHAR</td>
<td>255</td>
<td>Default language of the node</td>
</tr>
<tr>
<td>replyPolicy</td>
<td>VARCHAR</td>
<td>15</td>
<td>Policy that defines whether owners or publisher should receive replies to items</td>
</tr>
<tr>
<td>associationPolicy</td>
<td>VARCHAR</td>
<td>15</td>
<td>Policy that defines who may associate leaf nodes with a collection</td>
</tr>
<tr>
<td>maxLeafNodes</td>
<td>NUMBER</td>
<td>n/a</td>
<td>Max number of leaf nodes that a collection node might have</td>
</tr>
</tbody>
</table>
## 6.2 Openfire MySQL database creation script

```sql
# $Revision: 1650 $
# $Date: 2005-07-20 00:18:17 -0300 (Wed, 20 Jul 2005) $

CREATE TABLE ofUser (  
    username              VARCHAR(64) NOT NULL,  
    plainPassword         VARCHAR(32),  
    encryptedPassword     VARCHAR(255),  
    name                  VARCHAR(100),  
    email                 VARCHAR(100),  
    creationDate          CHAR(15) NOT NULL,  
    modificationDate      CHAR(15) NOT NULL,  
    PRIMARY KEY (username),  
    INDEX ofUser_cDate_idx (creationDate) ) ;

CREATE TABLE ofUserProp (  
    username              VARCHAR(64) NOT NULL,  
    name                  VARCHAR(100) NOT NULL,  
    propValue             TEXT NOT NULL,  
    PRIMARY KEY (username, name) ) ;

CREATE TABLE ofUserFlag (  
    username              VARCHAR(64) NOT NULL,  
    name                  VARCHAR(100) NOT NULL,  
    startTime             CHAR(15),  
    endTime               CHAR(15),  
    PRIMARY KEY (username, name),  
    INDEX ofUserFlag_sTime_idx (startTime),  
    INDEX ofUserFlag_eTime_idx (endTime) ) ;

CREATE TABLE ofPrivate (  
    username              VARCHAR(64) NOT NULL,  
    name                  VARCHAR(100) NOT NULL,  
    namespace             VARCHAR(200) NOT NULL,  
    privateData           TEXT NOT NULL,  
    PRIMARY KEY (username, name, namespace(100)) ) ;

CREATE TABLE ofOffline (  
    username              VARCHAR(64) NOT NULL,  
    messageID             BIGINT NOT NULL,  
    creationDate          CHAR(15) NOT NULL,  
    messageSize           INTEGER NOT NULL,  
    stanza                TEXT NOT NULL,  
    PRIMARY KEY (username, messageID) ) ;

CREATE TABLE ofPresence (  
    username              VARCHAR(64) NOT NULL,  
    offlinePresence       TEXT,  
    offlineDate           CHAR(15) NOT NULL,  
    PRIMARY KEY (username) ) ;

CREATE TABLE ofRoster (  
    rosterID              BIGINT NOT NULL,  
    username              VARCHAR(64) NOT NULL,  
    userPresence          TEXT,  
    userList             TEXT,  
    PRIMARY KEY (rosterID, username) ) ;
```

### Annexes

```sql
CREATE TABLE ofRoster(
    rosterID BIGINT NOT NULL,
    jid VARCHAR(1024) NOT NULL,
    sub TINYINT NOT NULL,
    ask TINYINT NOT NULL,
    recv TINYINT NOT NULL,
    nick VARCHAR(255),
    PRIMARY KEY (rosterID),
    INDEX ofRoster_unameid_idx (username),
    INDEX ofRoster_jid_idx (jid(255))
);

CREATE TABLE ofRosterGroups(
    rosterID BIGINT NOT NULL,
    rank TINYINT NOT NULL,
    groupName VARCHAR(255) NOT NULL,
    PRIMARY KEY (rosterID, rank),
    INDEX ofRosterGroup_rosterid_idx (rosterID)
);

CREATE TABLE ofVCard(
    username VARCHAR(64) NOT NULL,
    vcard MEDIUMTEXT NOT NULL,
    PRIMARY KEY (username)
);

CREATE TABLE ofGroup(
    groupName VARCHAR(50) NOT NULL,
    description VARCHAR(255),
    PRIMARY KEY (groupName)
);

CREATE TABLE ofGroupProp(
    groupName VARCHAR(50) NOT NULL,
    name VARCHAR(100) NOT NULL,
    propValue TEXT NOT NULL,
    PRIMARY KEY (groupName, name)
);

CREATE TABLE ofGroupUser(
    groupName VARCHAR(50) NOT NULL,
    username VARCHAR(100) NOT NULL,
    administrator TINYINT NOT NULL,
    PRIMARY KEY (groupName, username, administrator)
);

CREATE TABLE ofID(
    idType INTEGER NOT NULL,
    id BIGINT NOT NULL,
    PRIMARY KEY (idType)
);

CREATE TABLE ofProperty(
    name VARCHAR(100) NOT NULL,
    propValue TEXT NOT NULL,
    PRIMARY KEY (name)
);

CREATE TABLE ofVersion(
    name VARCHAR(50) NOT NULL,
    version INTEGER NOT NULL,
    PRIMARY KEY (name)
);
```
CREATE TABLE ofExtComponentConf (    subdomain VARCHAR(255) NOT NULL,    wildcard TINYINT NOT NULL,    secret VARCHAR(255),    permission VARCHAR(10) NOT NULL,    PRIMARY KEY (subdomain)    );

CREATE TABLE ofRemoteServerConf (    xmppDomain VARCHAR(255) NOT NULL,    remotePort INTEGER,    permission VARCHAR(10) NOT NULL,    PRIMARY KEY (xmppDomain)    );

CREATE TABLE ofPrivacyList (    username VARCHAR(64) NOT NULL,    name VARCHAR(100) NOT NULL,    isDefault TINYINT NOT NULL,    list TEXT NOT NULL,    PRIMARY KEY (username, name),    INDEX ofPrivacyList_default_idx (username, isDefault)    );

CREATE TABLE ofSASLAuthorized (    username VARCHAR(64) NOT NULL,    principal TEXT NOT NULL,    PRIMARY KEY (username, principal(200))    );

CREATE TABLE ofSecurityAuditLog (    msgID BIGINT NOT NULL,    username VARCHAR(64) NOT NULL,    entryStamp BIGINT NOT NULL,    summary VARCHAR(255) NOT NULL,    node VARCHAR(255) NOT NULL,    details TEXT,    PRIMARY KEY (msgID),    INDEX ofSecurityAuditLog_tstamp_idx (entryStamp),    INDEX ofSecurityAuditLog_uname_idx (username)    );

# MUC Tables

CREATE TABLE ofMucService (    serviceID BIGINT NOT NULL,    subdomain VARCHAR(255) NOT NULL,    description VARCHAR(255),    isHidden TINYINT NOT NULL,    PRIMARY KEY (subdomain),    INDEX ofMucService_serviceid_idx (serviceID)    );

CREATE TABLE ofMucServiceProp (    serviceID BIGINT NOT NULL,    name VARCHAR(100) NOT NULL,    propValue TEXT NOT NULL,    PRIMARY KEY (serviceID, name)    );

CREATE TABLE ofMucRoom (    serviceID BIGINT NOT NULL,    roomID BIGINT NOT NULL,
creationDate CHAR(15) NOT NULL,
modificationDate CHAR(15) NOT NULL,
name VARCHAR(50) NOT NULL,
naturalName VARCHAR(255) NOT NULL,
description VARCHAR(255),
lockedDate CHAR(15) NOT NULL,
emptyDate CHAR(15) NULL,
canChangeSubject TINYINT NOT NULL,
maxUsers INTEGER NOT NULL,
publicRoom TINYINT NOT NULL,
moderated TINYINT NOT NULL,
membersOnly TINYINT NOT NULL,
canInvite TINYINT NOT NULL,
roomPassword VARCHAR(50) NULL,
canDiscoverJID TINYINT NOT NULL,
logEnabled TINYINT NOT NULL,
subject VARCHAR(100) NULL,
rolesToBroadcast TINYINT NOT NULL,
useReservedNick TINYINT NOT NULL,
canChangeNick TINYINT NOT NULL,
canRegister TINYINT NOT NULL,

PRIMARY KEY (serviceID,name),
INDEX ofMucRoom_roomid_idx (roomID),
INDEX ofMucRoom_serviceid_idx (serviceID)
)

CREATE TABLE ofMucRoomProp (
roomID BIGINT NOT NULL,
name VARCHAR(100) NOT NULL,
propValue TEXT NOT NULL,
PRIMARY KEY (roomID, name)
)

CREATE TABLE ofMucAffiliation ( roomId BIGINT NOT NULL, jid TEXT NOT NULL, affiliation TINYINT NOT NULL, PRIMARY KEY (roomId,jid(70))
)

CREATE TABLE ofMucMember ( roomId BIGINT NOT NULL, jid TEXT NOT NULL, nickname VARCHAR(255) NULL, firstName VARCHAR(100) NULL, lastName VARCHAR(100) NULL, url VARCHAR(100) NULL, email VARCHAR(100) NULL, faqentry VARCHAR(100) NULL, PRIMARY KEY (roomId,jid(70))
)

CREATE TABLE ofMucConversationLog ( roomId BIGINT NOT NULL, sender TEXT NOT NULL, nickname VARCHAR(255) NULL, logTime CHAR(15) NOT NULL, subject VARCHAR(255) NULL, body TEXT NULL, INDEX ofMucConversationLog_time_idx (logTime)
)

# PubSub Tables
CREATE TABLE ofPubsubNode (  
serviceID VARCHAR(100) NOT NULL,  
nodeID VARCHAR(100) NOT NULL,  
leaf TINYINT NOT NULL,  
creationDate CHAR(15) NOT NULL,  
modificationDate CHAR(15) NOT NULL,  
parent VARCHAR(100) NULL,  
deliverPayloads TINYINT NOT NULL,  
maxPayloadSize INTEGER NULL,  
persistItems TINYINT NULL,  
maxItems INTEGER NULL,  
notifyConfigChanges TINYINT NOT NULL,  
notifyDelete TINYINT NOT NULL,  
notifyRetract TINYINT NOT NULL,  
presenceBased TINYINT NOT NULL,  
sendItemSubscribe TINYINT NOT NULL,  
publisherModel VARCHAR(15) NOT NULL,  
subscriptionEnabled TINYINT NOT NULL,  
configSubscription TINYINT NOT NULL,  
accessModel VARCHAR(10) NULL,  
payloadType VARCHAR(100) NULL,  
bodyXSLT VARCHAR(100) NULL,  
dataformXSLT VARCHAR(100) NULL,  
creator VARCHAR(255) NOT NULL,  
description VARCHAR(255) NULL,  
language VARCHAR(255) NULL,  
name VARCHAR(50) NULL,  
replyPolicy VARCHAR(15) NULL,  
associationPolicy VARCHAR(15) NULL,  
maxLeafNodes INTEGER NULL,  
PRIMARY KEY (serviceID, nodeID)  
)
;

CREATE TABLE ofPubsubNodeJIDs (  
serviceID VARCHAR(100) NOT NULL,  
nodeID VARCHAR(100) NOT NULL,  
jid VARCHAR(255) NOT NULL,  
associationType VARCHAR(20) NOT NULL,  
PRIMARY KEY (serviceID, nodeID, jid(70))  
)
;

CREATE TABLE ofPubsubNodeGroups (  
serviceID VARCHAR(100) NOT NULL,  
nodeID VARCHAR(100) NOT NULL,  
rosterGroup VARCHAR(100) NOT NULL,  
INDEX ofPubsubNodeGroups_idx (serviceID, nodeID)  
)
;

CREATE TABLE ofPubsubAffiliation (  
serviceID VARCHAR(100) NOT NULL,  
nodeID VARCHAR(100) NOT NULL,  
jid VARCHAR(255) NOT NULL,  
affiliation VARCHAR(10) NOT NULL,  
PRIMARY KEY (serviceID, nodeID, jid(70))  
)
;

CREATE TABLE ofPubsubItem (  
serviceID VARCHAR(100) NOT NULL,  
nodeID VARCHAR(100) NOT NULL,  
id VARCHAR(100) NOT NULL,  
jid VARCHAR(255) NOT NULL,  
creationDate CHAR(15) NOT NULL,  
PRIMARY KEY (serviceID, nodeID, id)  
)
payload    MEDIUMTEXT NULL,
            PRIMARY KEY (serviceID, nodeID, id)
        );

CREATE TABLE ofPubsubSubscription (
    serviceID   VARCHAR(100) NOT NULL,
    nodeID      VARCHAR(100) NOT NULL,
    id          VARCHAR(100) NOT NULL,
    jid         VARCHAR(255) NOT NULL,
    owner       VARCHAR(255) NOT NULL,
    state       VARCHAR(15)   NOT NULL,
    deliver     TINYINT       NOT NULL,
    digest      TINYINT       NOT NULL,
    digest_frequency    INT       NOT NULL,
    expire      CHAR(15)      NULL,
    includeBody VARCHAR(30)   NULL,
    subscriptionType VARCHAR(10)   NOT NULL,
    subscriptionDepth TINYINT   NOT NULL,
    keyword     VARCHAR(200)  NULL,
    PRIMARY KEY (serviceID, nodeID, id)
        );

CREATE TABLE ofPubsubDefaultConf (
    serviceID VARCHAR(100) NOT NULL,
    leaf       TINYINT       NOT NULL,
    deliverPayloads TINYINT       NOT NULL,
    maxPayloadSize INTEGER       NOT NULL,
    persistItems TINYINT       NOT NULL,
    maxItems    INTEGER       NOT NULL,
    notifyConfigChanges TINYINT       NOT NULL,
    notifyDelete TINYINT       NOT NULL,
    notifyRetract TINYINT       NOT NULL,
    presenceBased TINYINT       NOT NULL,
    sendItemSubscribe TINYINT       NOT NULL,
    publisherModel VARCHAR(15)   NOT NULL,
    subscriptionEnabled TINYINT       NOT NULL,
    accessModel     VARCHAR(10)   NOT NULL,
    language       VARCHAR(255)  NULL,
    replyPolicy    VARCHAR(15)   NULL,
    associationPolicy VARCHAR(15)   NOT NULL,
    maxLeafNodes   INTEGER       NOT NULL,
    PRIMARY KEY (serviceID, leaf)
        );

# Finally, insert default table values.

INSERT INTO ofID (idType, id) VALUES (18, 1);
INSERT INTO ofID (idType, id) VALUES (19, 1);
INSERT INTO ofID (idType, id) VALUES (23, 1);
INSERT INTO ofID (idType, id) VALUES (26, 2);
INSERT INTO ofVersion (name, version) VALUES ('openfire', 21);
# Entry for admin user
INSERT INTO ofUser (username, plainPassword, name, email, creationDate,
    modificationDate)
    VALUES ('admin', 'admin', 'Administrator', 'admin@example.com', '0',
    '0');
# Entry for default conference service
INSERT INTO ofMucService (serviceID, subdomain, isHidden) VALUES (1,
    'conference', 0);
6.3 Installation instructions for middleware clients

Intro

This section outlines the installation instructions of the middleware clients. These embed components developed by different consortium partners and developed in different programming languages, e.g. the IEC 61850 protocol stack is developed in python while the rest of the sub-components in Java.

Given the interface difficulties between these two languages, a dedicated Java to Python bridge was developed to facilitate seamless information exchange between the sub-components. This bridge started out as a standalone module, but during the course of development its implementation embedded the additional OS4ES client sub-components. So effectively the bridge comprises the entire functionality of the OS4ES client. The name “Java Python bridge” remained in the documentation due to legacy reasons.

The Java Python bridge aims to the translation of the hat-open-dist stack to a Java program.

Dependencies

Applications:

- Python >= 3.5
- Java version >= 1.8
- Thrift-py >= 0.3.7

Windows development machine setup

Install Python and Java correct versions and follow the installation instructions of hat-open-dist stack. If a IEC 61850 Server needs to run at your machine install thrift py server by following the next steps:

- unzip thriftpy-0.3.7.tar.gz available in /thriftPyServer folder
- go to /dist folder
- unzip again thriftpy-0.3.7.tar
- Using the cmd with Administrator rights go to the last unzipped folder
- execute "python setup.py install"

In order to run the JavaPyhtonBridge from a development environment:

1. If you want to run an IEC 61850 Server a) define iec61850_server.yaml file configuration in /thriftPyServer folder & b) run the thrift-py server by double clicking
1. On the /thriftPyServer/thriftPyServerRun.bat file in order to initialize the thrift-py server.

2. If you want to run an IEC 61850 Client a) define the iec61850_client.yaml file configuration in the JavaPythonBridge folder. b) copy client/os4es_iec61850_client.py file to the installation_python_path/Lib folder.

3. The /hat folder along with its files should be included to installation_python_path/Lib/site-packages folder.

4. The libraries in the the /lib folder should be included in the project libraries dependencies.

An example class simulating the same tests as the hat stack test file is provided at the examples source package.

**Linux/Ubuntu operating systems**

Download hat-open-dist and JavaPythonBridge repository files locally and execute the following commands for correct installation of the stack:

**Update and upgrade system**

- $ sudo apt-get upgrade
- $ sudo apt-get update

**Install Java 8:**

- $ sudo add-apt-repository ppa:webupd8team/java
- $ sudo apt-get update
- $ sudo apt-get install oracle-java8-installer

**Install Python3.5.1:**

- $ sudo apt-get install libssl-dev openssl
- $ wget https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz
- $ tar xzvf Python-3.5.1.tgz
- $ cd Python-3.5.1
- $ ./configure --enable-shared LDFLAGS="-L/usr/local/lib"
- $ sudo make
- $ sudo make install

**Install pip3.5 and hat-open-dist stack pip requirements:**

- $ sudo pip install --upgrade pip
- $ cd @hat-pen-dist_path
- $ sudo pip3.5 install -r requirements.pip.txt

Copy hat folder to python3.5 site-packages folder:
In case of running an IEC 61850 Server install thrift-py, if not skip this step:

- $ sudo pip3.5 install thriftpy

In case of running an IEC 61850 Client copy os4es_iec61850_client.py to the installation_python_path folder, if not skip this step:

- $ cd @JavaPythonBridge_path
- $ cp client/os4es_iec61850_client.py /usr/local/lib/python3.5/

In order to run the JavaPythonBridge from a development environment:

1. If you want to run an IEC 61850 Server
   - a) define iec61850_server.yaml file configuration in the /thriftPyServer folder
   - b) run thrift-py server by executing the following commands:
     - $ cd @JavaPythonBridge_path/thriftPyServer
     - $ python3.5 thrift_server.py &
     - for shutting down the thrift-py server run:
       - $ ps aux|grep thrift
       - $ sudo kill -9 thrift-server.py_process_id

2. If you want to run an IEC 61850 Client
   - a) define the iec61850_client.yaml file configuration in the JavaPythonBridge folder.
   - b) when running Java program include JavaPythonBridge_path and /usr/local/lib path at the Java library path
     - example: $ sudo java -jar -Djava.library.path=/home/superuser/OS4ES/JavaPythonBridge

3. The libraries in the /lib folder should be included in the project libraries dependencies.

Documentation

Documentation is available in /doc folder.
6.4 Middleware client API descriptions

6.4.1 IEC 61850 client - Class Os4esIec61850Client

java.lang.Object

gr.hypertech.os4es.core.Os4esIec61850Client

public class Os4esIec61850Client
extends java.lang.Object

Field Summary

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifier and Type</td>
</tr>
<tr>
<td>java.util.List&lt;ReportListener&gt;</td>
</tr>
</tbody>
</table>

Constructor Summary

<table>
<thead>
<tr>
<th>Constructors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor and Description</td>
</tr>
<tr>
<td>Os4esIec61850Client ()</td>
</tr>
</tbody>
</table>

Method Summary

<table>
<thead>
<tr>
<th>All Methods</th>
<th>Instance Methods</th>
<th>Concrete Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifier and Type</td>
<td>Method and Description</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>connect_client (int client_id) establish client connection with id In case of OSI stack communication, configuration is specified by 'hat://drivers/iec61850/client.yaml'</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>create_client (java.lang.String clientConfigPath) create a new iec61850 client and get the client-id</td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>delete_client (int client_id) delete iec61850 client with id</td>
<td></td>
</tr>
<tr>
<td>java.lang.String</td>
<td>delete_dataset (int client_id, java.lang.String[] ds_refs) iec 61850-7-2 SC 13 deletedataset</td>
<td></td>
</tr>
</tbody>
</table>
### void disconnect_client(int client_id)
Close client connection with id

### void final_library()
Finalizes the native library

### void get_report(int clientId, java.lang.String methodName, java.lang.String signature, int reportTimeout)
If a report is available a java method will be called.

### void init_library()
Initializes the native library

iec 61850-7-2 SC 20 cancel

### java.lang.String py_confirm_edit_sg_values(int client_id, java.lang.String logical_device, int data)
iec 61850-7-2 SC 16 set_edit_sg_value COROUTINE - Confirm editing of the Setting Group.

COROUTINE - Get buffered report control block values
Returns value of brcb data attribute

Returns type description of data object or data attribute object
COROUTINE - Get data directory

Native call to get_data_directory

Returns the value of data object or data attribute object

### java.lang.String[] py_get_dataset_directory(int client_id, java.lang.String ds_logical_device,
| Returns list of dataset entries in dataset object (each entry is represented as ordered list [LD, LN, FC, DO,...]  
|  
| COROUTINE - Get data set directory  
|  
| java.lang.String | **py get logical device directory** (int client_id, java.lang.String logical_device)  
| Native call to get_logical_device_directory  
|  
| java.lang.String[] | **py get logical node directory** (int client_id, java.lang.String logical_device, java.lang.String logical_node, java.lang.String acsi_class)  
| client get_logical_node_directory COROUTINE - Get logical node directory  
|  
| java.lang.String[] | **py get server directory** (int client_id)  
| Native call to get_server_directory  
|  
| java.lang.String | **py get sgcb values** (int client_id, java.lang.String logical_device)  
| iec 61850-7-2 SC 16 getsgcbvalues COROUTINE - Read SGCB attribute values.  
|  
| COROUTINE - Get unbuffered report control block values  
| Returns value of urcb data attribute  
|  
| iec 61850-7-2 SC 20 operate  
|  
| java.lang.String | **py select active sg** (int client_id, java.lang.String logical_device, int data)  
| iec 61850-7-2 SC 16 selectactivesg COROUTINE - Select active setting group.  
|  
| java.lang.String | **py select edit sg** (int client_id, java.lang.String logical_device, int data)  
| iec 61850-7-2 SC 16 selecteditsg COROUTINE - Select editing setting group.  
|  
### Annexes

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iec 61850-7-2 SC 20 selectwithvalue</code></td>
<td></td>
</tr>
<tr>
<td><code>py_set_edit_sg(int client_id, java.lang.String logical_device, java.lang.String logical_node, java.lang.String path, java.lang.String data)</code></td>
<td>Sets the value of data object or data attribute object.</td>
</tr>
</tbody>
</table>
Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

reportListeners

public java.util.List<ReportListener> reportListeners

Constructor Detail

Os4esIec61850Client

public Os4esIec61850Client()

Method Detail

init_library

public void init_library()
Initializes the native library

final_library

public void final_library()
Finalizes the native library

create_client

public int create_client(java.lang.String clientConfigPath)
create a new iec61850 client and get the client-id
Parameters:
clientConfigPath - the path to the client configuration file
Returns:
the just created clientId

delete_client

public void delete_client(int client_id)
delete iec61850 client with id
Parameters:
client_id - the id of the client to delete

connect_client

public void connect_client(int client_id)
establish client connection with id In case of OSI stack communication, configuration is specified by 'hat://drivers/iec61850/client.yaml#'
client_id - the id of the client to be connected

disconnect_client

public void disconnect_client(int client_id)

close client connection with id

Parameters:
client_id - client id

py_get_server_directory

public java.lang.String[] py_get_server_directory(int client_id)

Native call to get_server_directory

Parameters:
client_id - client Id

Returns:
the list of logical nodes

py_get_logical_device_directory

public java.lang.String[] py_get_logical_device_directory(int client_id,
java.lang.String logical_device)

Native call to get_logical_device_directory

Parameters:
client_id - client id
logical_device - logical device

Returns:
the list of logical nodes

py_get_logical_node_directory

public java.lang.String[] py_get_logical_node_directory(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String acsi_class)

client get_logical_node_directory COROUTINE - Get logical node directory

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
logical_node - the logical's device logical node string
acsi_class - (AcsiClass) acsi services class class

Returns:
Returns list of `acsi_class` instance names on logical node
Return: `acsi_class` instance names (list[string]) success None
error Returns if `acsi_class` is DATA_SET then returned value is list of data set names (List[str]) List[LogicalNodeEntry]: if `acsi_class` is not DATA_SET then returned value is list of logical node entries None: on error class

```
<table>
<thead>
<tr>
<th>py_get_data_directory</th>
</tr>
</thead>
</table>
| public java.lang.String[] py_get_data_directory(int client_id,
| java.lang.String logical_device,
| java.lang.String logical_node,
| java.lang.String data_object,
| java.lang.String functional_constraint) |
```

Native call to get_data_directory

Parameters:
- client_id - client id
- logical_device - logical device
- logical_node - logical node
- data_object - data object
- functional_constraint - functional constraint

Returns:
the data object directory

```
<table>
<thead>
<tr>
<th>py_get_data_definition</th>
</tr>
</thead>
</table>
| public java.lang.String py_get_data_definition(int client_id,
| java.lang.String logical_device,
| java.lang.String logical_node,
| java.lang.String data_object,
| java.lang.String logical_device,
| java.lang.String data_object_element, |
| java.lang.String functional_constraint) |
```

Returns type description of data object or data attribute object COROUTINE - Get data directory

Parameters:
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
data_object - the logical's node data object

data_object_element - data object element (data object or data attribute)

functional_constraint - (str) functional constraint

Returns:

Return: type description (string) example: return: {Vstring255, Vstring255, Vstring255, Vstring255} or {Bool, {Byte, OString64}, Ubyte, Utctime, Bool, BVstring2} or {Ulong, Ulong, {{Float}, {Float}, {Float}, {Float}, {Float}}} or {Bool, BVstring13, Utctime} or OString64 etc. Returns type description Return type

Optional[hat.drivers.mms.common.TypeDescriptionBase] class

hat.drivers.iec61850.client.Reader(conn) Bases: object class

hat.drivers.mms.common.types.TypeDescriptionBase Bases: object

Base class for all type descriptions

py_get_data_values

public java.lang.String py_get_data_values(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String functional_constraint,
java.lang.String path)

Returns the value of data object or data attribute object

Parameters:

client_id - the id of the client

logical_device - the client's logical device string

logical_node - the logical's device logical node string

functional_constraint - (str) functional constraint

path - (List[str]) single item path ([DO, ..., DA, ...])

Returns:

Return: data value (string) Type:

Optional[hat.drivers.mms.common.DataBase]

py_set_data_values

public java.lang.String py_set_data_values(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String functional_constraint,
java.lang.String path,
java.lang.String data)

Set the value of data object or data attribute object

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
logical_node - the logical's device logical node string
functional_constraint - (str) - functional constraint
path - (List[str]) single item path ([DO, ..., DA, ...])
data - (DataBase) data value

Returns:
Return: write result (string) Optional[SetDataValuesResult]
example: logical_device: AA1E1Q1FP1OC4_1 logical_node: LPHD1
fc: ST path: PhyHealth value: {1,0000000000000,1970-01-01 01:00:00.0[00000000]} class

`py_get_dataset_directory`

```java
public java.lang.String[] py_get_dataset_directory(int client_id,
                                                java.lang.String ds_logical_device,
                                                java.lang.String logical_node,
                                                java.lang.String data_set)
```

Returns list of dataset entries in dataset object (each entry is represented as ordered list [LD, LN, FC, DO,...] COROUTINE - Get data set directory

Parameters:
client_id - the id of the client
ds_logical_device - the client's logical device string
logical_node - the logical's device logical node string
data_set - data set name

Returns:
Return: dataset entries (list[string]) success None error
ReturnType: Optional[List[List[str]]]

`create_dataset`

```java
public java.lang.String create_dataset(int client_id,
                                        java.lang.String ds_logical_device,
                                        java.lang.String ds_logical_node,
                                        java.lang.String ds_name,
                                        java.lang.String[] ds_members )
```

iec 61850-7-2 SC 13 createdataset

Parameters:
client_id - the id of the client
ds_logical_device - data set logical device
ds_logical_node -- data set logical node

ds_name -- data set name

ds_members - (List[List[str]]) -- list of data set members where each member is represented as ordered list ([LD, LN, FC, DO, ...])

Returns:

delete_dataset

public java.lang.String delete_dataset(int client_id, java.lang.String[] ds_refs)

Parameters:
client_id - the id of the client
ds_refs -- list of data set references represented as ordered list ([LD, LN, ds_name])

Returns:
result (string)

py_get_urcb_values


COROUTINE - Get unbuffered report control block values Returns value of urcb data attribute

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
logical_node - the logical's device logical node string
urcb -- unbuffered report control
urcb_da -- data attribute

Returns:
Return: data attributes value (string) - Return type: Optional[hat.drivers.mms.common.DataBase] class hat.drivers.mms.common.data.DataBase Bases: object Base class for all data

py_set_urcb_values

public java.lang.String py_set_urcb_values(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String urcb,
java.lang.String urcb_da,
                java.lang.String value)

set the value of urcb data attribute (use for configuration only)

**Parameters:**
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
- urcb - unbuffered report control
- urcb_da - data attribute
- value - the value to set

**Returns:**
Return: write result (string)

```java
public java.lang.String py_get_brcb_values(
    int client_id,
    java.lang.String logical_device,
    java.lang.String logical_node,
    java.lang.String brcb,
    java.lang.String brcb_da)
```

**COROUTINE** - Get buffered report control block values

**Returns value of brcb data attribute**

**Parameters:**
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
- brcb - buffered report control block
- brcb_da - data attribute

**Returns:**
- data attributes value (string) - Return type:
  - Optional[hat.drivers.mms.common.DataBase] class
  - hat.drivers.mms.common.data.DataBase Bases: object Base class for all data

```java
public java.lang.String py_set_brcb_values(
    int client_id,
    java.lang.String logical_device,
```
set the value of urcb data attribute (use for configuration only)

Parameters:
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
- brcb - buffered report control block
- brcb_da - data attribute
- data - (DataBase) data value

Returns:
- write result (string) Optional[SetDataValuesResult]

```
get_report

public void get_report(int clientId,
                        java.lang.String methodName,
                        java.lang.String signature,
                        int reportTimeout)
```

If a report is available a java method will be called. Therefore the 2 parameter, first the name of the method, second the signature. The method need the following parameter to call:
- rpt_id (string) - opt_flds (string) - seq_num (int) - time_of_entry (string) - data_set (string) - buf_ovfl (int) - entry_id (string) - conf_rev (int) - sub_sq_num (int) - more_segments_follow (int) - data_values (list of string) - data_refs (list of string) - data_reasons (list of string) According to the iec 61850 reporting model only the elements set in the opt_flds bitstring as true are relevant.

Parameters:
- clientId - the id of the client
- methodName - the name of the method
- signature - the report signature
- reportTimeout - the timeout in seconds that the report waiting should expire

```
reportCallback

public void reportCallback(java.lang.String jrpt_id,
                           java.lang.String jopt_flds,
                           int sq_num,
                           java.lang.String jtime_of_entry,
                           java.lang.String jdata_set,
                           int buf_ovfl,
                           java.lang.String jentry_id,
                           int conf_rev,
                           int sub_sq_num,
```
int more_follow,
java.lang.String[] values,
java.lang.String[] reasons,
java.lang.String[] refs)

The Java callback method that is fed with the next report object when the get_report native function is called Parameters of this callback function are the report object attributes

Parameters:
- jrpt_id - report report id
- jopt_flds - report option fields
- sq_num - report sequence number
- jtime_of_entry - report timestamp
- jdata_set - report dataset name
- buf_ovfl - report buffer overflow flag
- jentry_id - report entry id
- conf_rev - report configuration revision
- sub_sq_num - report subsequence number
- more_follow - report more segments follow
- values - result
- reasons - report data reason codes
- refs - report data reason references

```java
public java.lang.String py_get_sgcb_values(int client_id,
java.lang.String logical_device)
```

**iec 61850-7-2 SC 16 getsgcbvalues COROUTINE - Read SGCB attribute values.**

Parameters:
- client_id - the id of the client
- logical_device - the client's logical device string

Returns:
- data value (string) Return type: Optional[DataBase] class
- hat.drivers.mms.common.data.DataBase Bases: object Base class for all data Return example should be as follows. Example StructureData({ UnsignedData, # NumOfSG UnsignedData, # ActSG UnsignedData, # EditSG BooleanData, # CnfEdit UtcTimeData, # LActTm IntegerData # ResvTms (optional) })

```java
public java.lang.String py_get_edit_sg_value(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String functional_constraint,
```
Annexes

java.lang.String path

iec 61850-7-2 SC 16 geteditsgvalue COROUTINE - Read value(s) from Setting Group.

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
logical_node - the logical's device logical node string
functional_constraint - (str) - functional constraint ('SG' or 'SE')
path - (List[str]) single item path (List[str]) - single item path (['DO', ..., 'DA', ...])

Returns:
data value (string) returnType Optional[DataBase] class
hat.drivers.mms.common.data.DataBase Bases: object Base class for all data

```java
py_select_active_sg
```

public java.lang.String py_select_active_sg(int client_id,
java.lang.String logical_device,
int data)

iec 61850-7-2 SC 16 selectactivesg COROUTINE - Select active setting group.

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
data - (UnsignedData) setting group number as unsigned integer size 8

Returns:
write result (string) Return type:
Optional[SetDataValuesResult] class
hat.drivers.iec61850.client.SetDataValuesResult Bases:
enum.Enum An enumeration.

```java
py_select_edit_sg
```

public java.lang.String py_select_edit_sg(int client_id,
java.lang.String logical_device,
int data)

iec 61850-7-2 SC 16 selecteditsg COROUTINE - Select editing setting group.

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
data - (UnsignedData) setting group number as unsigned integer size 8

Returns:
write result (string) Return type:
Optional[SetDataValuesResult] class
hat.drivers.iec61850.client.SetDataValuesResult Bases:
enum.Enum An enumeration.

```java
public java.lang.String py_set_edit_sg(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String path,
java.lang.String data)
```

**py_set_edit_sg**

Parameters:
client_id - the id of the client
logical_device - the client's logical device string
logical_node - the logical's device logical node string
path - (List[str]) single item path (List[str]) - single item path ([DO, ..., DA, ...])
data - (DataBase) data value

Returns:
write result (string)

```java
public java.lang.String py_confirm_edit_sg_values(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
da
```
iec 61850-7-2 SC 20 select COROUTINE - Command select before operate without value.

Parameters:
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
- data_object - command data object

Returns:
- control object reference (string) - Return type: Optional[DataBase] Note: Return type should be Optional[VisibleStringData] but it’s server dependant. class hat.drivers.mms.common.data.DataBase Bases: object Base class for all data class hat.drivers.mms.common.data.VisibleStringData Bases: hat.drivers.mms.common.data.VisibleStringData, hat.drivers.mms.common.data.DataBase VisibleString data value str value

```java
py_select_with_value
```

```java
public java.lang.String py_select_with_value(int client_id,
                                             java.lang.String logical_device,
                                             java.lang.String logical_node,
                                             java.lang.String data_object,
                                             java.lang.String sbo_value)
```

iec 61850-7-2 SC 20 selectwithvalue

Parameters:
- client_id - the id of the client
- logical_device - the client's logical device string
- logical_node - the logical's device logical node string
- data_object - command data object
- sbo_value - data (DataBase) SBOw block data value

Returns:
- write result (string) example: data_object: AA1E1Q1P1LD0/SXCBR1.BlkCls oper_value: {False,{0,00},0,1970-01-01 01:00:00.0[00000000],False,00}

```java
py_operate
```

```java
public java.lang.String py_operate(int client_id,
                                    java.lang.String logical_device,
                                    java.lang.String logical_node,
                                    java.lang.String data_object,
                                    java.lang.String oper_value)
```
iec 61850-7-2 SC 20 operate

**Parameters:**
- `client_id` - the id of the client
- `logical_device` - the client's logical device string
- `logical_node` - the logical's device logical node string
- `data_object` - data object
- `oper_value` - operate value

**Returns:**
Return: write result (string) example: `data_object: AA1E1Q1FP1LD0/SXCBR1.BlkCls oper_value: {False,{0,00},0,1970-01-01_01:00:00.0[00000000],False,00}`

```
public java.lang.String py_cancel(int client_id,
java.lang.String logical_device,
java.lang.String logical_node,
java.lang.String data_object,
java.lang.String cancel_value)
```

iec 61850-7-2 SC 20 cancel

**Parameters:**
- `client_id` - the id of the client
- `logical_device` - the client's logical device string
- `logical_node` - the logical's device logical node string
- `data_object` - the data object to be selected
- `cancel_value` - cancel value

**Returns:**
write result (string) example: `data_object: AA1E1Q1FP1LD0/SXCBR1.BlkCls oper_value: {False,{0,00},0,1970-01-01_01:00:00.0[00000000],False,00}`
6.4.2 IEC 61850 server - Class Os4esIec61850Server

java.lang.Object

gr.hypertech.os4es.core.Os4esIec61850Server

```java
public class Os4esIec61850Server
    extends java.lang.Object
```

**Constructor Summary**

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Os4esIec61850Server()</td>
</tr>
</tbody>
</table>

**Method Summary**

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>close()</td>
</tr>
<tr>
<td>void</td>
<td>createServer()</td>
</tr>
<tr>
<td>void</td>
<td>getData(java.lang.String logicalDevice, java.util.List&lt;java.lang.String&gt; address)</td>
</tr>
<tr>
<td>Iec61850Data</td>
<td>getIecData()</td>
</tr>
<tr>
<td>void</td>
<td>setData(java.lang.String logicalDevice, java.util.List&lt;java.lang.String&gt; address, Iec61850Data data)</td>
</tr>
</tbody>
</table>

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait

**Constructor Detail**

```java
public Os4esIec61850Server() |
```
Method Detail

`createServer`

```java
public void createServer()
```

`'COROUTINE'` - Create IEC 61850 server configuration file already handled by the thrift implementation

`close`

```java
public void close()
```

`setData`

```java
public void setData(java.lang.String logicalDevice,
                    java.util.List<java.lang.String> address,
                    Iec61850Data data)
```

Set data

Parameters:

- logicalDevice - logical device
- address - item's address
- data - data

`getData`

```java
public Iec61850Data getData(java.lang.String logicalDevice,
                            java.util.List<java.lang.String> address)
```

Get data

Parameters:

- logicalDevice - logical device
- address - item's address

Returns:

- data value
References

[9] “OS4ES Description of Work”.