Deliverable 3.1
Analysis of Smart Grid communication requirements, protocol model development and technological mappings evaluation
D3.1 Analysis of Smart Grid communication requirements, protocol model development and technological mappings evaluation

Document Information

<table>
<thead>
<tr>
<th>Programme</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>D3.1</td>
</tr>
<tr>
<td>WP / Task related</td>
<td>WP 3 / Task 3.1 and 3.2</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>01/08/2015</td>
</tr>
<tr>
<td>Status and Version</td>
<td>Final version 1.5</td>
</tr>
<tr>
<td>Number of pages</td>
<td>57 pages</td>
</tr>
<tr>
<td>Document Responsible</td>
<td>Stjepan Sučić – Končar</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Stjepan Sučić – Končar</td>
</tr>
<tr>
<td></td>
<td>Markus Breuers – FGH</td>
</tr>
<tr>
<td></td>
<td>Andrea Schröder – FGH</td>
</tr>
<tr>
<td></td>
<td>Robert Wagner – T-Systems MMS</td>
</tr>
<tr>
<td></td>
<td>Ronny Eckardt – T-Systems MMS</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Christoph Brunner – IT4POWER</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>16.02.2015</td>
<td>Stjepan Sučić</td>
<td>Draft with Table of Content</td>
</tr>
<tr>
<td>0.2</td>
<td>10.06.2015</td>
<td>Markus Breuers</td>
<td>Parts on SOAP and WebSockets (in German)</td>
</tr>
<tr>
<td>0.3</td>
<td>15.06.2015</td>
<td>Andrea Schröder</td>
<td>Translation of SOAP and WebSockets part</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Formatting of figures and tables</td>
</tr>
<tr>
<td>0.4</td>
<td>23.06.2015</td>
<td>Markus Breuers</td>
<td>Parts on device registration/deregistration, device search and security concepts of SOAP and WebSockets (in German)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Modification of figures and tables</td>
</tr>
<tr>
<td>0.5</td>
<td>01.07.2015</td>
<td>Andrea Schröder</td>
<td>Translation of device registration / deregistration, device search and security concepts of SOAP and WebSockets</td>
</tr>
<tr>
<td>0.6</td>
<td>03.07.2015</td>
<td>Ronny Eckardt</td>
<td>Parts on SIP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Extending Abbreviations for all chapters</td>
</tr>
<tr>
<td>0.7</td>
<td>09.07.2015</td>
<td>Robert Wagner</td>
<td>Parts about Security and Privacy</td>
</tr>
<tr>
<td>0.8</td>
<td>23.07.2015</td>
<td>Stjepan Sučić</td>
<td>Final version</td>
</tr>
</tbody>
</table>
Executive Summary

This deliverable provides analysis of Smart Grid communication requirements defined in D2.1 in order to develop protocol model that fulfils OS4ES platform requirements. Furthermore, several middleware technologies have been analysed in order to find the suitable technical approach for protocol implementation:

- Section 1 provides an introduction and shows the scope of this document.
- Section 2 describes the protocol development framework, protocol service functionalities and protocol data model requirements.
- Section 3 provides an overview of technological mapping candidates and middleware selection criteria.
# Table of contents

Document Information........................................................................................................... 1
Revision History..................................................................................................................... 2
Executive Summary............................................................................................................... 3
Table of contents.................................................................................................................. 4
List of Tables........................................................................................................................ 7
List of Figures....................................................................................................................... 8

1 Introduction......................................................................................................................... 9
  1.1 Scope of the document ................................................................................................. 9
  1.2 Notations, abbreviations and acronyms ..................................................................... 9

2 Protocol model – development framework, service functionalities and data model ....... 12
  2.1 ACSI overview ........................................................................................................... 12
  2.2 ACSI classes notation ............................................................................................... 13
  2.3 ACSI services ............................................................................................................ 14
  2.4 Mapping D2.1 Application-level protocol functions to IEC 61850 ACSI ............... 15
  2.5 Device registration and deregistration ACSI ............................................................ 16
    2.5.1 Via IEC 61850 reporting model .......................................................................... 16
    2.5.2 Via reason for IEC 61850 association ................................................................ 20
    2.5.3 Via IEC 61850 data model ................................................................................ 22
  2.6 Device search ACSI .................................................................................................... 23
  2.7 Contract management ............................................................................................... 24
  2.8 Access rights management ACSI .............................................................................. 24

3 Technological mapping ....................................................................................................... 25
  3.1 Specific Communication Service Mapping ............................................................... 25
  3.2 Mapping principles .................................................................................................... 25
  3.3 IEC 62541 (OPC UA) - [IEC 62541] ..................................................................... 26
    3.3.1 Technology overview ......................................................................................... 26
    3.3.2 Object model mapping ...................................................................................... 28
    3.3.3 Information security features ............................................................................. 29
    3.3.4 Open source implementations ............................................................................ 30
  3.4 IEC 61400-25-4 Annex A ......................................................................................... 30
3.4.1 Introduction – technology overview ................................................................. 30
3.4.2 Object model mapping ....................................................................................... 31
3.4.3 Mapping solicited services ................................................................................ 32
3.4.4 Mapping of unsolicited services ...................................................................... 32
3.4.5 Information security .......................................................................................... 33
3.4.6 Open source implementations .......................................................................... 34

3.5 Devices Profile for Web Services (DPWS) ............................................................. 35
3.5.1 Introduction – technology overview ................................................................. 35
3.5.2 Object model Mapping ...................................................................................... 36
3.5.3 Mapping solicited services ................................................................................ 37
3.5.4 Mapping of unsolicited services ...................................................................... 37
3.5.5 Existing implementations .................................................................................. 39
3.5.6 Information security features .......................................................................... 39
3.5.7 Open source implementations .......................................................................... 40

3.6 XML messaging over WebSocket ........................................................................... 40
3.6.1 Introduction – technology overview ................................................................. 40
3.6.2 Object model mapping ...................................................................................... 41
3.6.3 Mapping solicited services ................................................................................ 41
3.6.4 Mapping of unsolicited services ...................................................................... 42
3.6.5 Information security .......................................................................................... 43
3.6.6 Open source implementations .......................................................................... 43

3.7 XMPP (Extensible Messaging and Presence Protocol) .......................................... 44
3.7.1 Introduction – technology overview ................................................................. 44
3.7.2 Object model mapping ...................................................................................... 45
3.7.3 Mapping solicited services ................................................................................ 45
3.7.4 Mapping of unsolicited services ...................................................................... 46
3.7.5 Information security .......................................................................................... 46
3.7.6 Open source implementations .......................................................................... 46

3.8 Session Initiation Protocol (SIP) ......................................................................... 47
3.8.1 Introduction – technology overview ................................................................. 47
3.8.2 Transport Types ............................................................................................... 48
3.8.3 SIP Messages ..................................................................................................... 50
3.8.3 Object model mapping................................................................. 51
3.8.4 Mapping solicited services ....................................................... 52
3.8.5 Mapping of unsolicited services ................................................. 52
3.8.6 Information security................................................................. 53
3.8.7 Open source implementations................................................. 53
3.9 Mapping comparison and comments............................................ 54

References.......................................................................................... 55
List of Tables

Table 1 : Acronyms list ........................................................................................................ 11
Table 2 : ACSI class definition [1] .................................................................................. 13
Table 3 : ACSI models and services [2]............................................................................ 14
Table 4 : Application-level communication protocol functions ....................................... 15
Table 5 : IEC 61850 Server-Associate ............................................................................ 17
Table 6 : IEC 61850 Server-Release ................................................................................ 19
Table 7 : IEC 61850 Server-Associate with the parameter ReasonForAssociation ............ 21
Table 8 : ACSI search and example queries ....................................................................... 24
Table 9 : Mapping solicited services .................................................................................. 24
Table 10 : HTTP methods .................................................................................................. 37
Table 11 : Example of three different synchronous IEC 61850 requests ......................... 42
Table 12 : Mapping comparison ........................................................................................ 54
List of Figures

Figure 1 : ACSI communication methods [1] ................................................................. 13
Figure 2 : Initial registration via IEC 61850 reporting model ........................................ 16
Figure 3 : Registry updates via IEC 61850 reporting model ........................................... 18
Figure 4 : Deregistration via IEC 61850 reporting model ................................................ 19
Figure 5 : Registration via IEC 61850 association model ................................................ 20
Figure 6 : Deregistration via IEC 61850 association model ............................................ 21
Figure 7 : Registration/Deregistration via IEC 61850 data model ..................................... 22
Figure 8 : DER System search ...................................................................................... 23
Figure 9 : ACSI mapping to communication stacks/profiles [1] ......................................... 25
Figure 10 : OPC UA Address Space components ............................................................ 27
Figure 11 : Modelling IEC 61850 data with OPC UA AddressSpace components .......... 28
Figure 12 : Relationships among IEC 61850 ACSI models and OPC UA service sets ....... 29
Figure 13 : SOAP message ......................................................................................... 30
Figure 14 : HTTP „Waiting for an event” ........................................................................ 32
Figure 15 : HTTP „Callback” ........................................................................................ 33
Figure 16 : WS – Security concept .................................................................................. 34
Figure 17 : DPWS stack overview .................................................................................. 35
Figure 18 : The DPWS entities and ACSI services ......................................................... 36
Figure 19 : DPWS mapping – WSDL file structure ......................................................... 36
Figure 20 : WS-Eventing specification entities ............................................................... 38
Figure 21 : Event-driven data exchange ....................................................................... 38
Figure 22 : Asynchronous communication via WebSockets ......................................... 43
Figure 23 : Architecture main choices ......................................................................... 44
1 Introduction

1.1 Scope of the document

This deliverable defines functionalities and data models requirements of the protocol model services required by the OS4ES platform. Several middleware technologies have been analysed in order to find the most appropriate approach for implementing protocol. The selected middleware technology has been mapped to the specific protocol requirements.

1.2 Notations, abbreviations and acronyms

The following table lists all notations, abbreviations and acronyms that are used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACSI</td>
<td>Abstract Communication Service Interface</td>
</tr>
<tr>
<td>ASN.1</td>
<td>Abstract Syntax Notation One</td>
</tr>
<tr>
<td>BER</td>
<td>Basic Encoding Rules</td>
</tr>
<tr>
<td>DA</td>
<td>Data Attribute</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DO</td>
<td>Data Object</td>
</tr>
<tr>
<td>DPWS</td>
<td>Devices Profile for Web Services</td>
</tr>
<tr>
<td>FC</td>
<td>Functional Constraint</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>GOOSE</td>
<td>Generic Object Oriented System Event</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ICT</td>
<td>information and communication technologies</td>
</tr>
<tr>
<td>IEC</td>
<td>International Electro-technical Commission</td>
</tr>
<tr>
<td>IED</td>
<td>Intelligent Electronic Device</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IP</td>
<td>Internet protocol</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunication Union (agency of the United Nations)</td>
</tr>
<tr>
<td>JID</td>
<td>Jabber Identifier</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>MMS</td>
<td>Manufacturing Message Specification of ISO-9506</td>
</tr>
<tr>
<td>OPC</td>
<td>Open Platform Communication</td>
</tr>
<tr>
<td>OPC UA</td>
<td>Open Platform Communication Unified Architecture</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>RCB</td>
<td>Report Control Block</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RPC</td>
<td>Remote Procedure Call</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>SASL</td>
<td>Simple Authentication and Security Layer protocol</td>
</tr>
<tr>
<td>SCSM</td>
<td>specific communication service mapping</td>
</tr>
<tr>
<td>SDP</td>
<td>Session Description Protocol</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>TPAA</td>
<td>Two Party Application Association</td>
</tr>
<tr>
<td>W3C</td>
<td>Wide Web Consortiums</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
</tbody>
</table>
### Table 1: Acronyms list

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XER</td>
<td>XML Encoding Rules</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>VOIP</td>
<td>Voice over IP</td>
</tr>
</tbody>
</table>
2 Protocol model – development framework, service functionalities and data model

Developing protocol model is based on defining technology-neutral functionalities and data model requirements required by communicating entities. This approach is thoroughly analysed in international standard IEC 61850 and is also known as Abstract Communication Service Interface (ACSI) [1]. Therefore, IEC 61850 ACSI principles have been used for developing OS4ES protocol model.

2.1 ACSI overview

According to the IEC 61850 standard, ACSI are defined by using an object-oriented modelling technique of communication services. The ACSI service interface uses an abstract modelling method what means that the definition is focused on the description of what the services provide. The concrete messages (and their encoding) to be exchanged between devices (how the services are built) are defined separately. This abstraction allows various mappings appropriate for different requirements and following state-of-the-art in communication technology without a need to change the protocol model.

The ACSI defines common utility services for energy automation devices. The two groups of communication services are depicted in Figure 1 [1]. One group uses a client-server model with services like control or get data values. A second group comprises a peer-to-peer model with services used for time-critical purposes (e.g. fast and reliable transmission of data between protection IEDs, from one IED to many remote IEDs) and with the sampled value services for transmissions based on a periodic basis.

Real clients and servers can be connected by a variety of communication systems. Communication media may have geographic and utilisation constraints, such as limited bit rates, proprietary data link layers, restricted times for use, and satellite hop delays. Systems may be hierarchical, with a few central sites authorising and managing the interactions with a large number of field sites, or it may be networked with peer-to-peer interactions. Communication media may have varying configurations, such as point to multi-point, multi-drop, meshed, hierarchical, WAN-to-LAN, intermediate nodes acting as routers, as gateways, or as data concentrator databases, etc.
2.2 ACSI classes notation

IEC 61850-7-2 uses the class notation as depicted in Table 2.

<table>
<thead>
<tr>
<th>ABC class</th>
<th>Attribute name</th>
<th>Attribute type</th>
<th>Value/value range/explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Attribute1 [1..n]</td>
<td>Type1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attribute2 [0..n]</td>
<td>Type2</td>
<td></td>
</tr>
<tr>
<td>Services</td>
<td>Service1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Service2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The attributes of a class shall have an attribute name and an attribute type. The multiplicity of the attributes are:

- [0..n] – attribute may be available zero to n times (for example dataSet [0..n] in the GenLogicalNodeClass)
- [1..n] – attribute may be available one to n times (for example DataObject [1..n] in the GenLogicalNodeClass)
- [0..1] – attribute may be available zero or one times (for example SettingGroupControlBlock [0..1] in the GenLogicalNodeClass)
## 2.3 ACSI services

Table 3 lists the ACSI service models and services supported by IEC 61850.

<table>
<thead>
<tr>
<th>Service model</th>
<th>Description</th>
<th>Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Represents the external visible behaviour of a device. All other ACSI models are part of the server.</td>
<td>GetServerDirectory</td>
</tr>
<tr>
<td>Application association</td>
<td>Provision of how two devices can be connected. Provides different views to a device: restricted access to the server's information and functions.</td>
<td>Associate, Abort, Release</td>
</tr>
<tr>
<td>Logical device</td>
<td>Represents a group of functions; each function is defined as a logical node.</td>
<td>GetLogicalDeviceDirectory</td>
</tr>
<tr>
<td>Logical node</td>
<td>Represents a specific function of the substation system, for example, overvoltage protection.</td>
<td>GetLogicalNodeDirectory, GetAllDataValues</td>
</tr>
<tr>
<td>Data</td>
<td>Provides a means to specify typed information, for example, position of a switch with quality information, and timestamp.</td>
<td>GetDataValues, GetDataDefinition, GetDataDirectory</td>
</tr>
<tr>
<td>Data set</td>
<td>Allow to group various data together.</td>
<td>GetDataSetValues, SetDataSetValues, CreateDataSet, DeleteDataSet, GetDataSetDirectory</td>
</tr>
<tr>
<td>Setting group control</td>
<td>Defines how to switch from one set of setting values to another one and how to edit setting groups.</td>
<td>SelectActiveSG, SelectEditSG, ConfirmEditSG, GetEditSGValues, GetSGCBValues</td>
</tr>
<tr>
<td>Reporting and logging</td>
<td>Describes the conditions for generating reports and logs based on parameters set by the client. Reports may be triggered by changes of process data values (for example, state change or deadband) or by quality changes. Logs can be queried for later retrieval. Reports may be sent immediately or deferred (buffered). Reports provide change-of-state and sequence-of-events information exchange.</td>
<td>Buffered RCB: Report, GetBRCBValues, SetBRCBValues. Unbuffered RCB: Report, GetURCBValues, SetURCBValues. Log CB: GetLCBValues, SetLCBValues. QueryLogByTime, QueryLogAfter, GetLogStatusValues</td>
</tr>
<tr>
<td>Generic substation events</td>
<td>Provides fast and reliable system-wide distribution of data; peer-to-peer exchange of IED binary status information. GOOSE means Generic Object Oriented Substation Event and supports the exchange of a wide range of possible common data organised by a DATA-SET</td>
<td>GOOSE CB: SendGOOSEMessage, GetGoReference, GetGOOSEElementNumber, GetGoCBValues, SetGoCBValues</td>
</tr>
<tr>
<td>Transmission of sampled values</td>
<td>Fast and cyclic transfer of samples, for example, of instrument transformers.</td>
<td>Multicast SVC: SendMSVMessage, GetMSVCBValues, SetMSVCBValues. Unicast SVC: SendUSVMessage, GetUSVCBValues, SetUSVCBValues</td>
</tr>
<tr>
<td>Control</td>
<td>Describes the services to control, for example, devices or parameter setting groups.</td>
<td>Select, SelectWithValue, Cancel, Operate, Command, CommandTermination, TimeActivatedOperate</td>
</tr>
<tr>
<td>Time and time synchronisation</td>
<td>Provides the time base for the device and system.</td>
<td>TimeSynchronization</td>
</tr>
<tr>
<td>File transfer</td>
<td>Defines the exchange of huge data blocks such as programs.</td>
<td>GetFile, SetFile, DeleteFile, GetFileAttributeValues</td>
</tr>
</tbody>
</table>

Table 3: ACSI models and services [2]
2.4 Mapping D2.1 Application-level protocol functions to IEC 61850 ACSI

The *X-function supported, empty-function not supported

Table 4 shows the results of the mapping process between IEC 61850 ACSI services and application-level protocol functions required by OS4ES platform [3]. It can be seen that the ACSI services:

- Device registration
- Device search
- Access rights management

are not supported by the existing IEC 61850 standard ACSI services. Therefore, for the purpose of the OS4ES platform, ACSI model of these services need to be developed.

<table>
<thead>
<tr>
<th>Service model</th>
<th>Device registration</th>
<th>Device search</th>
<th>Contract management</th>
<th>Access rights management</th>
<th>Control command / Setpoint change</th>
<th>Status update / Measurement update</th>
<th>Data model management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Application association</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Logical device</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical node</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Data set</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Setting group control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Reporting and logging</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Generic substation events</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission of sampled values</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Time and time synchronisation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>File transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*X-function supported, empty-function not supported

Table 4: Application-level communication protocol functions
2.5 Device registration and deregistration ACSI

Registering and deregistering a DER System in the OS4ES Registry by means of IEC 61850 communication can be realized in various ways:

- (De)Registration using the IEC 61850 reporting model
- (De)Registration using the IEC 61850 associating model or
- (De)Registration with the IEC 61580 data model.

The following sections 2.5.1 - 2.5.3 describe these possible registration options with the help of their communication pattern and the necessary IEC 61850 ACSI extensions.

2.5.1 Via IEC 61850 reporting model

This solution requires a RCB (Report Control Block) to be implemented in the DER System that is foreseen to be used by the OS4ES Registry. By means of the RCB the yellow pages of the OS4ES Registry can be updated.

![Diagram of initial registration via IEC 61850 reporting model]

Figure 2: Initial registration via IEC 61850 reporting model
Figure 2 shows the initial registration process, that is the first contact of a DER System with the OS4ES Registry. Thereby the DER System establishes an association to the OS4ES Registry. This process of a server-side initiation of an IEC 61850 association is not part of the current IEC 61850-7-2 (ACSI) that only defines a client-side association [2]. A new ACSI service for IEC 61850-7-2 would have the following parameters (Table 5):

<table>
<thead>
<tr>
<th>IEC 61850 ACSI Server-Associate Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
</tr>
<tr>
<td>ClientAccessPointReference</td>
</tr>
<tr>
<td>AuthenticationParameter</td>
</tr>
<tr>
<td>Response+</td>
</tr>
<tr>
<td>AssociationID</td>
</tr>
<tr>
<td>Result</td>
</tr>
<tr>
<td>Response-</td>
</tr>
<tr>
<td>ServiceError</td>
</tr>
</tbody>
</table>

Table 5: IEC 61850 Server-Associate

After the association has been confirmed by a positive response by the OS4ES Registry, the Registry itself becomes active and requests the white page data by means of the service GetDataValues. Then the (pre-configured) RCB with the help of which the publication of yellow page data is realized is switched on and a GI (General Interrogation) report is created which concludes the initial registration of a DER System.

Figure 3 shows the update process of the OS4ES Registry. Within the DER System the RCB reserved and activated by the OS4ES Registry is configured in such a way that a report is generated and sent to the OS4ES Registry every time data changes occur. So, the update process is triggered by an event (change of data) and is therefore fully automated. This implies that no actions of the Registry client are required. It is therefore definitely the most effective way of an update process.
For the deregistration of a DER System at the OS4ES Registry two different options are available using the IEC 61850 reporting model:

1. Deregistration is handled as a particular communication interaction offered by the DER System. This interaction can be done by notifying the OS4ES Registry using the reporting functionality. For accomplishing this it is either possible to define an additional variable Unregister or to extend the bit string featuring available services of a DER System. In any case the data set of the RCB needs to be configured accordingly.

2. A persistent IEC 61850 association is established. This is not necessarily required for option 1 as in this case the availability of service offers could be checked in fixed time intervals or an association could be kept as needed. If a persistent association between DER System and OS4ES Registry is assumed that deregistration would mean a release of this association. As this process is only defined for the client-side in the current version of IEC 61850-7-2 (ACSI) a server-side release would be needed as an extension to the standard. Figure 4 shows the communication scheme for deregistering a DER System via a server-side release.

Figure 3: Registry updates via IEC 61850 reporting model

For the deregistration of a DER System at the OS4ES Registry two different options are available using the IEC 61850 reporting model:

1. Deregistration is handled as a particular communication interaction offered by the DER System. This interaction can be done by notifying the OS4ES Registry using the reporting functionality. For accomplishing this it is either possible to define an additional variable Unregister or to extend the bit string featuring available services of a DER System. In any case the data set of the RCB needs to be configured accordingly.

2. A persistent IEC 61850 association is established. This is not necessarily required for option 1 as in this case the availability of service offers could be checked in fixed time intervals or an association could be kept as needed. If a persistent association between DER System and OS4ES Registry is assumed that deregistration would mean a release of this association. As this process is only defined for the client-side in the current version of IEC 61850-7-2 (ACSI) a server-side release would be needed as an extension to the standard. Figure 4 shows the communication scheme for deregistering a DER System via a server-side release.
Figure 4: Deregistration via IEC 61850 reporting model

The new ACSI service would be defined as shown in Table 6.

<table>
<thead>
<tr>
<th>IEC 61850 ACSI Server-Release Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
</tr>
<tr>
<td>AssociationID</td>
</tr>
<tr>
<td>Response+</td>
</tr>
<tr>
<td>Result</td>
</tr>
<tr>
<td>Response-</td>
</tr>
<tr>
<td>ServiceError</td>
</tr>
</tbody>
</table>

Table 6: IEC 61850 Server-Release
2.5.2 Via reason for IEC 61850 association

A realisation with the IEC 61850 Association model assumes that a DER System registering at a registry always sends the reason for its registration. The following reasons could apply: initial registration, data update, unregister. It would also be conceivable that an update is specified during the association process, that is, that the modification would be directly sent and would not require to be read explicitly. This depends on the implementation within the registry and the overall system, respectively. Figure 5 shows that a (general) reason is sent with the association request and that the OS4ES Registry obtains the data of the DER System by means of a GetDataValue request.

Like in the case of the IEC 61850 reporting model described in section 2.5.1, the ACSI of IEC 61850-7-2 would need to be extended by a server-side Associate service. The definition of this new service (see Table 7) would deviate of the one given in section 2.5.1 and would additionally include the parameter ReasonForAssociation.
Table 7: IEC 61850 Server-Associate with the parameter ReasonForAssociation

Figure 6 describes the deregistration process of a DER System. As it can be seen no additional service is needed except for the Release request.
2.5.3 Via IEC 61850 data model

Another option for registering and deregistering a DER System at a registry is presented in Figure 7.

![Diagram showing registration and deregistration via IEC 61850 data model](image)

Figure 7: Registration/Deregistration via IEC 61850 data model

This option is comparable with the option using the IEC 61850 Association model. The only difference is that the reason for association is not directly sent with the Associate service but in a dedicated Data Object (DO). As soon as the DER System has established an association by means of a server-side Associate (Table 5) this DO is read by the Registry as a basic principle. Depending on the content of this DO further services are executed. If an initial registration or an update shall be executed further data are read. In case of deregistration the DER System is deleted in the Registry and the association is released.
2.6 Device search ACSI

The OS4ES service device search for searching the OS4ES Registry for DER Systems with specific characteristics and/or services represents a non-existent IEC 61850 service that has to redesigned. The following schema depicts the communication processes of such a service within the scope of the OS4ES architecture.

Some examples for potential registry queries in the OS4ES project are formulated below:

1. Search for DER Systems that can provide in total the amount of 5MW active power at the next day between 8pm and 9.20pm (-> Virtual Power Plants).
2. Search for DER Systems in the network area X that are able to shift their consumption by 5 to 6 hours in time so that in the following hour an active power reduction of 500 kW can be achieved (-> Flexibility in time).
3. Search for DER Systems that can raise their active power production so that a total of additional 750 kW can be reached (-> Flexibility in energy corridor).

Possible search parameters are described in the OS4ES deliverable 4.2 - “Specification of DER Semantic Models and EMS Matching Algorithms”.

Figure 8: DER System search
### 2.7 Contract management

It was decided that the actual processes to engage in, revise, renew or terminate a contract between a Resource Provider and an Aggregator in order to provide a given service are out of scope of the OS4ES framework. It is assumed that such those process have been executed resulting in a valid contract to mandate an Aggregator using / controlling a DER Resource service.

It is assumed that it is the Flexibility Provider role, which represents the commercial relationships of an aggregator, is responsible for the correct execution of these processes and for the registration of the contract in the Registries (such as DER-Registry), although both parties will hold all the information related to the contract. This statement will be considered as a prerequisite required by the operational processes supported by the OS4ES framework.

The OS4ES framework supports both contracting of services beforehand as well as contracting 'on the fly' as a result of the exchange of information by means of the OS4ES system between the Flexibility provider and the DER Resource. The latter is possible as the only dependency of the contracting phase is the availability of some attributes needed to identify and access the service provided by the Resource Provider.

### 2.8 Access rights management ACSI

It was decided that access rights management functionality not be part of communication protocol and therefore, will not develop access rights management ACSI. This functionality will be solved by the application-specific feature.
3 Technological mapping

3.1 Specific Communication Service Mapping

The specific communication service mapping (SCSM) defines how the services and the models (server, logical devices, logical nodes, data, data sets, report controls, log controls, setting groups, etc.) are implemented using a specific communication stack, i.e. a complete profile [4]. The mappings and the used application layer define the syntax (concrete encoding) for the data exchanged over the network.

The concept of the SCSM has been introduced to be independent from communication stacks including application protocols. One objective of the IEC 61850 series is the interoperability of devices. This requires that all communicating devices use the same communication stack. Therefore, the goal of this independence is not to have many mappings in parallel, but to be able to follow the state of the art in communication technology.

According to Figure 9, the SCSM maps the abstract communication services, objects and parameters to the specific application layers. These application layers provide the concrete coding. Depending on the technology of the communication network, these mappings may have different complexities, and some ACSI services may not be supported directly in all mappings, but equivalent services shall be provided (see example below). An application layer may use one or more stacks (layer 1 to 6).

![Figure 9: ACSI mapping to communication stacks/profiles [1]](image)

3.2 Mapping principles

- Introduction – technology overview
- Object model mapping
- Mapping solicited services
- Mapping of unsolicited services
- Open source implementations
3.3 IEC 62541 (OPC UA) - [IEC 62541]

3.3.1 Technology overview

OPC Unified Architecture (OPC UA) is a state-of-the-art device-level SOA technology and a successor of one of the most influential technologies in the field of industrial automation, called Open Platform Communication (OPC) [5]. In contrast to OPC, which is based on Microsoft’s COM/DCOM, OPC UA is truly platform-independent and based on open standards [6]. Similar to the IEC 61850 concept, OPC UA defines a set of application-level services which are independent of implementation technology while equipped with information modelling capability.

3.3.1.1 OPC UA Services

These services are based on defining request and response parameters that are sent between the client and the server. OPC UA services are logically grouped into service sets that enable server discovery, session manipulation, address space management and, lastly, polling-based and event-driven data exchange. The services can be implemented as UA Web Services and UA Native services. UA Web Services are based on a set of WS-* specifications similar to DPWS as described in, while UA Native uses a simple binary network protocol which integrates certain security mechanisms and runs directly on top of TCP/IP [6].

Related Services are logically grouped into Service Sets as follows:

- **Discovery Service Set**: Services that allow a client to discover the server and its configuration.
- **SecureChannel Service Set**: Services that enable secure communication channel between client and server.
- **Session Service Set**: Services that allow the user (client side) authentication and managing Session (logical long-running connection between a client and a server).
- **NodeManagement Service Set**: Services that allow the client adding and deleting Nodes and References in the AddressSpace.
- **View Service Set**: Services for browsing the AddressSpace or its subsets (Views).
- **Query Service Set**: Allows clients to get a subset of data values from the AddressSpace.
- **Attribute Service Set**: Services that allow clients to read and write Attributes of Nodes, including the value attribute.
- **Method Service Set**: Services that enable clients to call Methods.
- **MonitoredItem Service Set**: Services that allow clients to create, modify, and delete MonitoredItems (items used to monitor Attributes for value changes).
- **Subscription Service Set**: Services that allow clients to create, modify and delete Subscriptions (collections of MonitoredItems which in case of value change send Notification to the client).

Depending on type of application, not all OPC UA Services have to be supported. Hence, several Profile Categories [7] are defined. Basic ones are: Client, Server, Security and Transport.
3.3.1.2 OPC UA AddressSpace

The second important feature of OPC UA is its meta-modelling mechanism based on the AddressSpace concept [8]. AddressSpace elements are fundamental in creating information models based on collections of semantic data which the OPC UA server provides to OPC UA clients.

The basic AddressSpace unit is a Node comprised of References and Attributes. References define its relationships with the other Nodes (e.g. node A contains node B), while Attributes define Node characteristics (e.g. unique node identification). All Nodes are instances of the BaseNode class. Nodes are divided into several NodeClasses which define metadata for AddressSpace components as shown in figure 3. These are as follows:

- **Object**: A Node representing a physical or abstract element of a system. An Object comprises Variables and Methods and allows relationships to other Objects.
- **ObjectType**: A Node representing the type definition of an Object.
- **Variable**: A Node containing a value. There are two types of variables, Properties (server-defined characteristics of Nodes) and DataVariables (content of an Object).
- **VariableType**: A Node representing the type definition of a Variable.
- **Method**: A callable software function that is a component of an Object. Similar to the class method in object-oriented programming.
- **DataType**: Defines the data type of a Variable.
- **ReferenceType**: A Node representing the type definition and semantics of a Reference. ReferencesTypes are divided into hierarchical (used for constructing data hierarchies) and non-hierarchical (used for defining non-hierarchical relationships).
- **View**: A specific subset of the AddressSpace that is of interest to the OPC UA client.

Classes Object, Variable and Method are key NodeClasses that enable object-oriented system design. Information model structuring is done with Objects, ReferenceTypes and Views, thus providing several information model hierarchies for the same OPC UA server.
### 3.3.2 Object model mapping

Mapping an OPC UA server to standardized DER semantics requires modelling IEC 61850 data classes with *AddressSpace Nodes*. In order to enable this, IEC 61850 data classes are divided into two groups, classes used for modelling data hierarchy and classes representing system endpoints. Besides Data Attribute (DA) model, all other IEC 61850 data classes are used for structuring data hierarchy. Hence, these classes are modelled as the *ObjectTypes* connected with *HasComponent* hierarchical references. *HasComponent* defines is-a-part-of relationship between two *Nodes*. DA models are presented as *VariableTypes* comprising one or more *DataTypes*. Since DAs represent current states and data values, they are modelled as *DataVariables*. Complex DAs can be realized as several *DataVariables* connected with *HasSubtype* reference type. *HasSubtype* defines subtype relationship between same *Node* types. Functional Constraints (FC) are integrated into model as *Property of ObjectType* representing Data Object (DO) referenced with *HasProperty*. DSs are modelled as specific *Views* in the *AddressSpace* that organize DO *Nodes* [9].

Figure 11 shows relationships among IEC 61850 data models classes and designed OPC UA *AddressSpace*.

![Figure 11: Modelling IEC 61850 data with OPC UA AddressSpace components](image)

Figure 11 also demonstrates that domain-specific OPC UA information model based on IEC 61850 semantics can be created by using only a subset of OPC UA modelling components. Integrating DER semantics into *AddressSpace* is first step in implementing VPP automation system based on OPC UA.

Despite being developed independently, both IEC 61850 ACSIs and OPC UA *Services* share common characteristics making them functionally equivalent. For example, *Report-Control-Block models* (RCBs) are most complex ACSIs. RCBs enable event-driven information exchange for vertical communication.
When using OPC UA as a mapping technology for IEC 61850, RCBs can be realized by combining MonitoredItem Service Set and Subscription Service Set. Substitution relationships among all distinctive services of IEC 61850 and OPC UA are shown in Figure 4 [10].

Figure 12 displays relationships between specific OPC UA Profile Category and IEC 61850 ACSI models. These relationships show that main concepts of both approaches are interchangeable at application level providing an exceptional groundwork for formal integration of OPC UA and IEC 61850 specifications.

### 3.3.3 Information security features

One essential mechanism to meet the security objectives is to establish a Secure Channel that is used to secure the communication between a client and a server. The Secure Channel provides encryption to maintain Confidentiality, Message Signatures to maintain Integrity and Digital Certificates to provide application Authentication for data that comes from the Application Layer and passes the secured data to the Transport Layer. The security mechanisms that are managed by the Communication Layer are provided by the Secure Channel. The security mechanisms provided by the Secure Channel services are implemented by a protocol stack that is chosen for the implementation [11].

The inherent security features provided by OPC UA technology are:

- **WS-SecureConversation [12]** is a Web Services specification, created by IBM and others, that works in conjunction with WS-Security, WS-Trust and WS-Policy to allow the creation and sharing of security contexts. Extending the use cases of WS-Security, the purpose of WS-SecureConversation is to establish security contexts for multiple SOAP message exchanges, reducing the overhead of key establishment.

- **UA Secure Conversation [11]** is a binary version of WS-Secure Conversation. It allows secure communication over transports that do not use SOAP or XML.
3.3.4 Open source implementations

There are several open source implementations of the OPC UA:

https://github.com/acplt/open62541/wiki/List-of-Open-Source-OPC-UA-Implementations

3.4 IEC 61400-25-4 Annex A

3.4.1 Introduction – technology overview

The fundamental structure of this solution for mapping IEC 61850 ACSI Services and objects is based on SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and HTTP (Hypertext Transfer Protocol) as transmission protocol. It corresponds with the Web Service standards of W3C (World Wide Web Consortiums) [13].

3.4.1.1 SOAP

SOAP is a network protocol by means of which data can be exchanged between systems and remote procedure calls can be performed. It does not impose any instructions for the semantic of application specific data to be sent but provides a framework allowing to transmit arbitrary application specific information in XML notation [14]. For sending messages any transport protocol can be used, e.g. FTP, SMTP, HTTP or JMS. In practice mostly HTTP is employed because of its compatibility with common network architectures (such as firewalls).

A SOAP message consists at least of an element called Envelope which needs to be assigned a local name. A mandatory child element of the Envelope is the Body element while the optional child element Header may precede the Body element. The Header may contain meta-data, e.g. for routing, for encoding or for the identification of transactions. The Body element holds the actual user data.

Below the typical XML structure of a SOAP message is shown:

```xml
<?xml version=1.0?>
<s:Envelope xmlns:s=http://www.w3.org/2003/05/soap-envelope>
  <s:Header>
  </s:Header>
  <s:Body>
  </s:Body>
</s:Envelope>
```
3.4.1.2 WSDL

The Web Services Description Language (WSDL) is a platform independent, programming language independent and protocol independent description language for Web Services in order to exchange messages based on XML [15]. WSDL is an industrial standard of the Wide Web Consortiums (W3C). It is a meta-language by means of which the offered functions, data, data types and exchange protocols of a Web Service can be described. It mainly defines operations that are accessible from outside as well as the parameters and return values of these operations. In detail, a WSDL document contains the following information:

- the interface
- the access protocol and details for deployment
- all necessary information for accessing the service in a machine-readable format

WSDL is often used in combination with SOAP and XML schema in order to offer Web Services in the Internet. A client requesting a Web Service can read the WSDL in order to see which functions are available for this Web Service on the server. All data types employed in the WSDL file are integrated in XML format. The source code necessary for composing the sent objects on the client side can be generated automatically from the WSDL file. The client can now use SOAP for invoking a function listed in the WSDL file.

3.4.1.3 HTTP

The Hypertext Transfer Protocol (HTTP) is a stateless protocol for transmitting data of the application layer via a computer network [16]. HTTP has been standardized by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C). By extensions of its request methods, header information and status code HTTP is not restricted to Hypertext but is increasingly used for exchanging arbitrary data.

The communication units in http between client and server are called messages. It is distinguished between two types of messages: requests from the client to the server and responses from the server to the client (sent upon a request). Each message consists of two parts: the message header (also named Header or HTTP Header) and the message body (also named Body). The header holds information about the body, e.g. used encodings or the type of content in order to allow the receiver to interpret it correctly. The body only contains the actual user data.

3.4.2 Object model mapping

In contrast to other approaches such as OPC-UA the IEC 61850 data model is not mapped to a data model contained in the used technology. The data model employed here mirrors exactly the IEC 61850 data model providing that the basic data types of the W3C are used for modelling the IEC 61850 data classes. Simple attributes are mapped to native XML. Structures are mapped as XML compositions.

The information model itself (IEC 61850-7-3 and 7-4) [17] [18] is not tackled explicitly in this context. Only the object names and object references are important with regard of the data
model and are used as possible parameters for service calls. The services themselves (and their parameters) are identical to those defined in IEC 61850-7-2.

### 3.4.3 Mapping solicited services

Solicited services as typical request/response communication pattern can be directly mapped via HTTP communication, because the HTTP protocol represents a straightforward request/response protocol. This means that first the client sends a request to the server before the server sends data (XML) to the client.

### 3.4.4 Mapping of unsolicited services

Although synchronous communication can be realized by this solution, asynchronous events are problematic. As the communication protocol HTTP requires a request from the client before the server can answer spontaneous server events can only be implemented with restrictions or via workarounds: Either the client waits for an event after it has subscribed at the server for receiving a request or a client-side and server-side infrastructure is implemented for realizing HTTP callbacks. Both options are shown in the following figures.

![Diagram](image)

**Figure 14**: HTTP „Waiting for an event"

Figure 14 shows a possible communication scheme for realizing asynchronous services, the Waiting for an event. The IEC 61850 client sends a subscribe request to the IEC 61850 server, e.g. activation of a report control block. The server does not directly answer this request. Instead the communication channel is kept open until an event occurs, e.g. DataChange. The resulting report is then published as response of the earlier request. After the client has received the report the mechanism can start again, the client sends again a request to the server in order to provide the server a possibility for sending any reports to the client.

It is obvious that this solution is of limited suitability in order to perform asynchronous client/server communication and that a system needs to be tailored according to it. E.g. it needs to be ensured that events occur continuously in time intervals that are not too big or that the server timeout for a client request is set to a very high value in order to make sure it is not discarded. In addition an open communication channel waiting for an event would be blocked and would therefore not allow any other communication. In total, this scenario should be seen as an island solution and not as a generally valid solution, independent of the overall architecture.
Another option to realize asynchronous communication by means of SOAP and HTTP is shown in Figure 15. The HTTP callback uses a two-part architecture: One communication channel is dedicated for subscribing for an event and another channel is used for the publication of an event. A necessary requirement for this approach is a dedicated SOAP server within the IEC 61850 client. This means additional implementation and network architecture effort. A possible scenario for event handling could be as follows:

The client sends a subscribe request to the server including his own WSDL for his SOAP Web Service. The client reacts to the subscription and can generate via the WSDL a corresponding event SOAP client suiting the server of the requesting client. The request is answered directly and the client is notified that for example the activation of the report control block was successful. If an event occurs at the server the resulting notification is directly sent as request from the SOAP client of the server to the SOAP server of the client. This means the processing of an event the IEC 61850 server takes over the role of a client while the IEC 61850 client acts as server.

To put it in a nutshell, the solutions proposed above represent possibilities to tackle event handling with IEC 61400-25-4 Annex A [19]. However, none of the two solutions is optimal. Asynchronous event handling is used in IEC 61850 for the reporting and control model – two important features for communication. So the conclusion is to neglect the non-optimal SOAP approach.

### 3.4.5 Information security

IEC 61400-25-4 Annex A [19] does not imply using any of the information security mechanisms. However, concerning security mechanisms SOAP as a classical Web Service can make use of the WS-Security Standard, a framework that addressing all relevant issues concerning Web Service Security.

WS-Security does not develop new procedures but combines existing security measures. It therefore forms a fundamental extension to the SOAP standards concerning security requirements. WS-Security [20] has been established by Microsoft, IBM and Verisign and is now further developed by OASIS.
Web Service Security defines a framework for embedding user and security information in a SOAP message. It does, however, not provide a ready-to-use solution for all security issues but provides the basis on which security specification can build.

The concept allows a broad range of security models such as SSL, SAML, Kerberos and Public Key Infrastructures (PKI). Moreover, WS-Security extends SOAP with encryption functions and signature functions based on XML encryption and XML signature, respectively.

Additionally WS-Security [20] allows for extensions that can help to solve special requirements, e.g.:

- **WS-Policy** (security measures and security restrictions)
- **WS-Trust** (framework for models of trust)
- **WS-Privacy** (data privacy)
- **WS-Secure-Conversation** (secure communication)
- **WS-Federation** (trust relationships in heterogeneous environments)
- **WS-Authorization** (authorization data and authorization guidelines)

![Figure 16: WS – Security concept](image)

### 3.4.6 Open source implementations

There are no available open source implementations of the IEC 61400-25-Annex A.
3.5 Devices Profile for Web Services (DPWS)

3.5.1 Introduction – technology overview

DPWS is a specific profile of Web Service protocols enabling SOA capabilities on resource constrained devices. It addresses SOA benefits such as service addressing, dynamic discovery, self-description and event notification mechanism for heterogeneous networked devices [21].

DPWS is partially based on Web Services Architecture (WSA) which enables creating highly-composeable and incrementally-integrated protocol stacks. The main difference from WSA is that DPWS does not require centralized service repository but it uses multicast service discovery instead. DPWS protocol stack uses several WS-* specification as shown in Figure 17.

These are as follows:

- **WS-Addressing**: Provides addressing mechanism for Web Services in transport-neutral manner. It decouples SOAP of underlying protocol and supports asynchronous message exchange.
- **WS-Discovery**: Discovery protocol based on UDP and multicast. It is used by devices to advertise themselves and discover other devices.
- **WS-MetadataExchange**: Defines data-types and operations for dynamic retrieval of device and services metadata, such as WSDL or XML schema definitions.
- **WS-Eventing**: Protocol for managing publish/subscribe mechanism and asynchronous event-driven data exchange.
- **WS-Security**: Flexible and feature-rich SOAP extension adding security to Web Services. It specifies how security, confidentiality and various security token formats can be integrated into SOAP messages.
- **WS-Policy**: Set of specifications that describe capabilities and constraints of various policies (e.g. security, Quality of Service) which can be advertised or required.

![Figure 17: DPWS stack overview](image)

DPWS middleware distinguishes two basic entities: *Hosting device* and *Hosted services*. The *Hosting device* is used for hosting one or several *Hosted services* which provide various customized application functionalities. The *Hosting device* supports also the built-in services of the DPWS stack, which allow dynamic device discovery (*WS-Discovery*), service self-description (*WS-MetadataExchange*) and event-driven data exchange (*WS-Eventing*). In order
to map IEC 61850 to DPWS, ACSI services need to be implemented as *Hosted services*. The relationships between ACSI services and DPWS entities are shown in Figure 18.

**Figure 18**: The DPWS entities and ACSI services

### 3.5.2 Object model Mapping

With SOAP Web Services, the services are mainly defined by a WSDL file. A WSDL file is made of five sections describing respectively (see Figure 19):

- *types*: XML Schema definitions used for the definition of the messages
- *message*: definition of the messages, based on types
- *portType*: operation definition in terms of input and output messages
- *binding*: definition of protocol and encoding associated to a portType
- *service*: assignment of real network addresses to bindings

**Figure 19**: DPWS mapping – WSDL file structure
3.5.3 Mapping solicited services

Most of the client/server services are solicited services defined by three elements: the Request, the positive Response (Response+) and the negative Response (Response-).

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>&lt;request parameters&gt;</td>
</tr>
<tr>
<td>Response+</td>
<td>&lt;response parameters&gt;</td>
</tr>
<tr>
<td>Response-</td>
<td>&lt;service errors&gt;</td>
</tr>
</tbody>
</table>

Table 9: Mapping solicited services...

In the DPWS mapping, these services use the Request/Reply message exchange pattern where the operations are defined in the WSDL by:

- One Input message corresponding to the Request
- One Output message corresponding to the Response+

The XML schema file defining the ASCI XML messaging is imported in the Types section of the WSDL. This schema defines all the elements required for definition of the wsdl:message elements. Then the definition of solicited services is quite straightforward in the WSDL file.

3.5.4 Mapping of unsolicited services

IEC 61850 reports: The RCB ACSI models specify application logic requirements for managing publish/subscribe mechanism and enabling event-driven data exchange [22]. IEC 61850 events are new DA values resulting from changed state of field devices (e.g. breaker trips) or reached threshold for supervised measurements. One or more events, together with related information, are grouped into IEC 61850 report and sent from the device towards the report handler application. IEC 61850 ACSI specifies two RCB class models, the buffered and unbuffered. The buffered RCB (BRCB) allows buffering events at device level and guarantees sequence-of-event functionality in case of loss of connection, while unbuffered RCB (URCB) issues immediate sending of reports on a best effort basis. RCB ACSI models involve three basic services ( [X]RCB represents BRCB or URCB):

- Set[X]RCBValues: Service used for managing settings of RCB object on server side. These settings enable/disable event-driven data exchange and define format of IEC 61850 reports.
- Get[X]RCBValues: Service used for retrieving settings of RCB object from server.
- Report: Service supporting the spontaneous data delivery mechanism.

DPWS events: Mapping IEC 61850 to DPWS requires implementing the RCB model application logic according to WS-Eventing specification. The publish/subscribe paradigm of WS-Eventing involves an Event source (publisher), a Subscriber, an Event sink and a Subscription manager.
WS-Eventing includes several subscription related operations: Subscribe (create subscription), Renew (update subscription expiration), GetStatus (retrieve subscription status), Unsubscribe (explicit deletion of subscription) and SubscriptionEnd (indication of unexpectedly ended subscription to an optional entity). The relationship among WS-Eventing entities and respective operations is depicted in Figure 20.

WS-Eventing only supports raw, application-neutral messages which do not encapsulate any topic information. Message notification is based on push as the default delivery mode.

Mapping notification mechanisms: The sequence diagram in Figure 21 outlines the proposed mapping solution for IEC 61850 and DPWS notification mechanisms.

![Figure 20: WS-Eventing specification entities](image)

From a modelling perspective the IEC 61850 application is the Event sink. In order to be consistent with the RCB ACSI, the Subscriber, Subscription manager and Event source are parts...
of the IEC 61850 device. This is possible since WS-Eventing allows decoupling the Event sink from the Subscriber.

Subscription process, on device side, starts after receiving a Set(X)RCBValues service request. The IEC 61850 device extracts the relevant SOAP message payload which is used for setting an RCB object. This payload is sent to Subscriber by calling a Mapping operation which is responsible for forming an adequate request at Subscriber. Then, the Subscriber sends a Subscribe request to the Subscription Manager (Event source). WS-Eventing notifications are IEC 61850 reports which are formatted according to the Set(X)RCBValues service request. Reports are forwarded directly from Event source to Event sink. Unsubscribe process has the same steps as the subscribe process except it forms adequate Unsubscribe request at Subscription Manager.

3.5.5 Existing implementations

Web Services for Devices (WS4D) [23] initiative was established by academic and industrial partners who were involved in European R&D SIRENA project. WS4D has resulted in several open source DPWS toolkits: WS4D-gSOAP (C/C++ based), WS4D-Axis2 (Java based), WS4D-JMEDS (Java based) and WS4D-uDPWS (C based).

Selection of WS4D DPWS toolkit depends on target platform and implementation specific features. Service Oriented Architecture for Devices (SOA4D) [24] is a similar project to WS4D initiated by Schneider Electric in 2007. It hosts two open source DPWS toolkits: DPWS Core (C based) and DPWS4J Core (Java based). Microsoft has also integrated the DPWS stack into its latest Windows versions (Vista and 7) in a form of Web Services on Devices API (WSDAPI) [25].

These implementations imply the fact that DPWS, opposite to MMS, has a large number of freely available, well supported and platform-neutral solutions. These benefits can significantly reduce time-to-market procedure for end products and also make DPWS a good candidate which could complement the current MMS mapping for streamlined and large-scale deployment of Smart Grid solutions based on IEC 61850 outside the substation automation domain.

3.5.6 Information security features

The inherent security features provided by DPWS technology is:

- **WS-Security**: An extension to SOAP to apply security to Web Services. It is a member of the Web Service specifications and was published by OASIS. The protocol specifies how integrity and confidentiality can be enforced on messages and allows the communication of various security token formats, such as Security Assertion Markup Language (SAML), Kerberos, and X.509. Its main focus is the use of XML Signature and XML Encryption to provide end-to-end security.
3.5.7 Open source implementations

There are several open source implementations of the DPWS

https://forge.soa4d.org/projects/dpwscore/
http://ws4d.e-technik.uni-rostock.de/

3.6 XML messaging over WebSocket

3.6.1 Introduction – technology overview

XML messaging over WebSocket is a combination of two network technologies. Via the WebSocket protocol a persistent client/server connection is established for the communication [26]. Messages to be transmitted are coded in XML format. In contrast to known Web Service solutions like SOAP, WSDL and the related RPC, that are all based on XML, the XML messages in this subchapter are based on Representational State Transfer (REST) paradigm.

3.6.1.1 WebSocket

The WebSocket protocol is an independent network protocol based on TCP. It has been designed to establish a bi-directional connection between a web application and a WebSocket server and a web server supporting WebSockets, respectively.

While in the case of a pure HTTP connection each action of the server requires a preceding request of the client, in the case of the WebSocket protocol it is sufficient if the client opens the connection. The server can then use this open connection actively and can provide new information to the client without having to wait for the client to provide a new connection.

At the start of each connection server and client perform a so called handshake. This resembles the HTTP header and is completely downwards compatible to it. This feature allows the use of the standard HTTP ports 80 (443 for TLS) both for normal HTTP communication and for the use of WebSockets. The following code shows an example handshake (request of a client to a server):

```
GET http://myHost/myService HTTP/1.1
Upgrade: websocket
Sec-WebSocket-Key: lnrIGb7MfU9aKplHKyZ5zA==
Sec-WebSocket-Version: 10
Sec-WebSocket-Protocol: soap
Accept-Encoding: gzip, deflate
Connection: Upgrade

HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: ROEP
```

After a positive response to this request that is interpreted by an HTTP server as a normal upgrade request, the HTTP connection is cancelled and is replaced by a WebSocket connection
with the same TCP/IP connection. So the communication channel between client and server is ready for use.

3.6.1.2 REST

REST designates a programming paradigm for distributed systems [27]. REST is an abstraction of the World Wide Webs (WWW) structure and behaviour. REST requires that a web address (URI) represents exactly the content of a page and that a Web-/REST-server answers to multiple requests with the same URI also with the same web site content.

The REST protocol has been developed of the HTTP Object Model designed by Roy Fielding in 1994. Fielding developed his idea of a unified concept further until he published the REST model as part of his dissertation Architectural Styles and the Design of Network-based Software Architectures in 2000.

In connection with REST websites and their content are regarded as resources. These resources are made available by a web server and can always unambiguously be identified by a URI.

In order to distinguish if a resource needs to be created, read, updated or deleted the following HTTP methods are used.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>Lists resources or gets a concrete resource</td>
</tr>
<tr>
<td>PUT</td>
<td>Updates a resource</td>
</tr>
<tr>
<td>DELETE</td>
<td>Deletes a resource or a collection of resources.</td>
</tr>
<tr>
<td>POST</td>
<td>The semantic may vary but usually a new resource is created.</td>
</tr>
</tbody>
</table>

Table 10: HTTP methods

In contrast to REST SOAP is no remote procedure call but comparable with the SQL commands INSERT, UPDATE, DELETE und SELECT.

3.6.2 Object model mapping

Neither the WebSockets protocol nor the REST protocol contain an object model of their own and can be regarded as pure communication technologies. This means that the transmission of data is the main purpose, not the data themselves. A concrete object model mapping can be discarded a priori. Because of the XML notation of the REST messages at best a mapping of objects in XML or JSON (JavaScript Object Notation) comes into question. This kind of mapping would then be considered completely independent of the WebSocket/REST approach described here.

3.6.3 Mapping solicited services

The normal case of an IEC 61850 server is a synchronous communication between server and client. That is the client sends a request to the server which in turn immediately sends a
response. In case of a positive response the response contains the requested data while in case of a negative response the response contains the respective failure message.

In the approach „XML messaging over WebSocket, that is described here, the association services which are inherently synchronous services take a special status. This has to do with the combination of two web technologies as mentioned before. Therefore the Association Model as described in IEC 61850-7-2 must be completely mapped to the WebSocket technology while all other models of 7-2 can be realized with REST like principles in XML notation.

Hence for the establishment of the connection the following is valid: A successful generation of a WebSocket between server and client can be equated with a positive response (Response+) Associate Service of IEC 61850-7-2 subchapter 8.3 and results in a TPAA (Two Party Application Association), a connection between client and server. The corresponding TPAA services Associate, Release and Abort must therefore always be used in the context of WebSockets.

All other synchronous services of IEC 61850-7-2 are realized with XML messages. The messages are similar to the normal REST communication, that is one of the HTTP methods listed in Table 10 are used with a specified URI. As no fixed connection (WebSocket) exists between server and client at the time of the service call neither the HTTP method (no HTTP communication takes place but instead WebSocket is used) nor the URI (IP address of the server ceases as no connection exists) can be applied unchanged. Consequently the HTTP-method represents the outer tag within the XML request and the URI without the IP address represents the IEC 61850 object reference. The following table shows three different examples of synchronous IEC 61850 requests.

<table>
<thead>
<tr>
<th>IEC61850 service</th>
<th>XML format</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDataValues</td>
<td>&lt;GET ref=chp01/LDCTRL1/LLN0/Mod fc=ST/&gt;</td>
</tr>
</tbody>
</table>
| SetDataValues          | <PUT ref=chp01/LDCTRL1/LLN0/Mod fc=ST>
|                        | <VALUE>1</VALUE></PUT>                        |
| DeleteDataSet          | <DELETE ref=chp01/LDCTRL1/LLN0/Dataset1/>       |

Table 11 : Example of three different synchronous IEC 61850 requests

The server would format the corresponding responses to the request in XML format and send them directly back to the client after processing the request.

3.6.4 Mapping of unsolicited services

Event handling as needed in the IEC 61850 world (e.g. for the Report Model and Control Model) can easily and effectively be realized with „XML messaging over WebSocket. The schema in Figure 22 describes such an asynchronous communication via WebSockets. In this case asynchronous means that a client request is not directly followed by a response. Instead the client subscribes (1) at the server for a certain event and is notified when the event occurs...
(2). Applied to e.g. the IEC 61850 Report Model the Subscribe/Unsubscribe process can be equalled with the switching on/off of a Report control block and the notification would represent the transmission of the actual report, triggered e.g. by a DataChange.

![Diagram of Asynchronous communication via WebSockets](image)

**Figure 22 : Asynchronous communication via WebSockets**

The parameter setting and activation (subscribe) of an event constitutes an easy, synchronous communication between client and server within which each request (e.g for modifying a value) is directly followed by a confirmation of the server. The requested control block in the server can be regarded as a self-contained resource in the sense of the REST paradigm and can therefore be configured with the methods PUT and/or POST shown in Table 10.

### 3.6.5 Information security

As a security mechanism for WebSockets the Transport Layer Security (TLS) protocol, a successor of Secure Sockets Layer (SSL) can be used. TLS provides protection of private data when communicating with users over the Internet. It makes sure that unauthorised third party can neither eavesdrop nor change the communication.

TLS has two layers, the TLS record protocol and the TLS handshake protocol. The TLS record protocol protects the association with methods like the Triple Data Encryption Standard (3DES) and can also be used without encryption. The TLS Handshake protocol provides a mutual authentication for server and clients and negotiates the applied encryption standards as well as the exchange of cryptographic keys before the data is transferred.

The TLS protocol is based on the SSL 3.0 protocol of Netscape. However, TLS and SSL are not interoperable. Therefore the TLS protocol contains a function which allows to degrade the connection in emergencies to SSL 3.0. Almost all current browsers support TLS at the time being. The protocol is further developed by the TLS Working Group.

Besides the transport security solution (TLS), a better protection level ensures IPSec/VPN.

### 3.6.6 Open source implementations

There are no available open source implementations of the XML messaging over WebSocket for IEC 61850 communication.
3.7 XMPP (Extensible Messaging and Presence Protocol)

3.7.1 Introduction – technology overview

XMPP is a communication protocol enabling two entities (XMPP clients) to exchange pieces of XML data, called stanzas. The XMPP clients are not directly connected together. They are instead connected to one or several intermediary entities (XMPP servers) making the routing of the stanzas [28].

Several servers can connect together in order to enable inter-domain communications between XMPP clients. The principle is that each client initiates a connection to its XMPP server (with TCP by default) and creates a logical channel called a stream. Servers connect together in a similar way so that a stanza sent by an XMPP client can flow through the server(s) up to the recipient XMPP client.

For each XMPP client, the permanent and implicitly outgoing connection consists in:

- Creating a TCP connection with the XMPP server (TLS may be used additionally)
- Authenticating through Simple Authentication and Security Layer (SASL)
- Opening two XMPP streams, one for upstream communication to the XMPP server, one for the downstream communication from the XMPP server
- Binding of a resource to the stream

Then, three kinds of stanzas can flow within a stream:

- iq dedicated for request / response exchanges – i.e. solicited services: <iq/>
- message dedicated for push exchanges – i.e. unsolicited services: <message/>
- presence dedicated for presence announcement: <presence/>

![Figure 23: Architecture main choices](image)
3.7.1.1 Address scheme

Each XMPP entity needs a unique address, called a Jabber Identifier (JID).

JID for entities look like email addresses – entity@domain.tld.

Every JID contains a domain portion and each entity that is foreseen to connect to a XMPP server will have a JID with a domain identifier that corresponds to the domain to which the XMPP server belongs. Static configuration or a DNS Service can be used by the XMPP entities to resolve the IP address of the XMPP server they are expected to connect to.

When an XMPP client is connected to a server it needs additionally to define a resource identifier for this connection. This can be the name of a device or software used on the client side to establish the connection or anything else. Then the full JID entity@domain.tld/resource identifies a given connection of an XMPP client.

3.7.1.2 Scalability and redundancy

In domains with huge number of members (i.e. XMPP clients), the XMPP server deployment can include a so called clustering, i.e. use of different physical devices to which the TCP connection requests will be initiated by the XMPP clients. In case of clustering, using DNS SRV records allows to balance the load of the different machines, as a weighted list of IP addresses of the connection managers (cluster) that can be involved. A weighted list can also be useful for managing path redundancy: in case the path between an XMPP client and its connection manager is interrupted, a connection to another connection manager, using therefore another path in the WAN, can be initiated by the XMPP client on its own initiative.

3.7.2 Object model mapping

Shall be per IEC 61850-8-1. The principle is to reuse the definitions of the part 8-1 regarding the mapping of the ACSI objects and services over the corresponding concepts of MMS. Then, instead of using the Basic Encoding Rules (BER) encoding for creating the Protocol Data Unit (PDU) like in the part 8-1, the new 61850-8-2 SCSM will use another ASN.1 encoding: the XML Encoding Rules (XER), defined by ITU X.693.

3.7.3 Mapping solicited services

All solicited services are implemented by IQ stanzas. The request message is implemented by an IQ stanza of type set or get and the response message is implemented by a ‘result’ IQ stanza. The direct child element of the IQ stanza, AssociationContext, represents the association established between the client and the server and it contains the payload generated by the presentation layer which includes the XML encoded MMS message (userData) embedded into end-to-end security information (SecurityRequest-PDU). The following example shows the IQ stanzas for the request and the positive response of a solicited service (using the A+ security profile).
3.7.4 Mapping of unsolicited services

With XMPP, the typical mechanism used to push information is the message stanza. So the unsolicited services are implemented as message stanzas. The direct child element of the stanza, AssociationContext, represents the association established between the client and the server and it contains a MMSpdu element which is the XML payload of the message. The following example shows a message stanza for the Report service.

3.7.5 Information security

XMPP belongs to the application layer in reference to the OSI layer model. TLS as a possible security solution can be used between XMPP Clients and the XMPP server. The specific threats countered at the transport layers level, as long as the XMPP server is a trusted hop, include:

- Unauthorized access to information through message level authentication and encryption of the messages
- Unauthorized modification (tampering) or theft of information through message level authentication and encryption of the messages

An additional simple authentication mechanism allows the XMPP server to guarantee the identity of the connected XMPP client. XMPP servers can be isolated (e.g., on a company intranet), and secure authentication (SASL) and encryption (TLS) have been built into the core XMPP specifications. XMPP was released in 1999. If an implementation of XMPP with TLS is preferred, some important security issues\(^1\) of the XMPP operators and developers should be integrated.

Besides the transport security solution (TLS) in XMPP, a better protection level ensures IPSec/VPN. With TLS it is not excluded, that the presence, the duration, the start and finishing time points could be recorded and evaluated from an attacker in the OS4ES network. If the attacker has access to a XMPP Server, it is necessary to implement End-To-End Encryption. A TLS solution must also ensure a check of the certificates. If not, there is no way to authenticate users, OS4ES clients and OS4ES XMPP.

3.7.6 Open source implementations

There are several open source implementations of the XMPP:

https://xmpp.org/xmpp-software/

\(^1\) https://github.com/stpeter/manifesto/blob/master/manifesto.txt, see 2015-07-09, Version 0.5
3.8 Session Initiation Protocol (SIP)

3.8.1 Introduction – technology overview

The Session Initiation Protocol (SIP) [29] was developed by the Internet Engineering Task Force (IETF). It was originally developed for voice applications in Internet Protocol (IP) based networks. Although from the beginning the protocol was envisioned for a much broader range of applications and for this reason designed for further extensibility in the future. Today SIP offers extensions i.e. for video conferencing, streaming multimedia distribution, instant messaging, presence information, file transfer, fax over IP and online games.

SIP is a text-based protocol, incorporating many elements of the Hypertext Transfer Protocol (HTTP) and the Simple Mail Transfer Protocol (SMTP) - but it is not compatible to these protocols.

SIP und its extensions are standardized in following RFCs:

- RFC 3261 – SIP
- RFC 3265 – SIP Extension: Specific Event Notification
- RFC 3515 – SIP Update: SIP Refer Method
- RFC 3665 – SIP Basic Call Flow Examples
- RFC 3581 – SIP Update: Symmetric Response Routing
- RFC 3853 – SIP Update: Using AES instead of 3DES
- RFC 4320 – SIP Update: Issues with the SIP Non-INVITE Transaction
- RFC 4916 – Connected Identity in the Session Initiation Protocol

SIP itself defines the messages that are sent between two or more endpoints for

- maintaining user registrations at a registrar
- setting up sessions
- defining session routing
- starting and controlling sessions
- handling various error scenarios
- modifying active sessions
- tearing down / terminating sessions

SIP sessions may consist of one or several real-time media streams – other streams are also possible.

For the negotiation of the media format and transport of the media data itself, SIP works in conjunction with several other application layer protocols, i.e.:

- Media identification and negotiation between communication partners is often achieved by embedding the Session Description Protocol (SDP) defined in RFC 4566.
For the transmission of media streams, like voice and video, SIP typically employs the Real-time Transport Protocol (RTP defined in RFC 3550) or Secure Real-time Transport Protocol (SRTP).

For example in VoIP:

- SIP is responsible to establish the session.
- The embedded SDP is used to negotiate Media details. During negotiation, the devices inform each other about the video and audio codecs they support, which protocol they want to support and on which network address they want to send and receive.
- Based on the result of the SDP negotiation, RTP may be finally used to transport the video and audio streams.
- The communication partners can control and change the RTP session using SIP commands.

3.8.1 Transport Types

SIP is an application layer protocol intended to be independent of the underlying transport layer. This enables the possibility to add new transport types that were originally not included in the standard, at the time when SIP itself was first defined. SIP mandates support for both UDP and TCP. Beside the later RFC 4158 defines SIP over SCTP.

SIP operates hop-by-hop – mostly clients do not communicate directly to each other, but rather use different network elements along the signalling path (please see chapter Network Elements). Each hop could have a different transport type. So a client may receive a message via TCP even if the original message was sent out via UDP.

SIP defines different behaviour per transport type only when the characteristics of the specific transport require it to do so. For example, UDP does not guarantee delivery, so SIP retransmits the messages. Using TCP, such retransmission is unnecessary, so no one should retransmit a message. This way the selection of the transport type has an impact on the behaviour of the transaction layer component.

For connection-based transports (e.g. TCP and SCTP), the state of the connections is maintained. Connections are kept open for an application-defined time and get reused to save time and resources. Because there is no defined limit for the number of different SIP messages that one can send on a connection, two devices such as proxies usually keep only one or a very few connections between them.

Note: Because of the clear separation of concerns SIP signalling messages always pass through the SIP server but media messages (i.e. RTP flow) can travel end-to-end between clients (User Agents) without passing through the SIP server.

Note: Transport types of the established sessions are independent of SIP transport types and depend on the protocols used for the session / streams.
3.8.1.1 Address scheme

Each resource of a SIP network is identified by a uniform resource identifier (URI) – like e-mails and WWW-servers [29]. The URI scheme used for SIP is sip: – in case of secure transmission the scheme sips: is used. Examples of URIs are:

- Unsecured SIP-connection: sip:user@domain
- Secure SIP-connection: sips:user@domain
- Telephone-number: tel:nummer (i.e. tel:+46-123-12345678909)

This last schema is used by devices offering gateways to the classic telecommunication network. It can be transferred into a SIP-URI by adding @domain (i.e. sip:+49-69-1234567@domain)

3.8.1.2 Architecture – Network elements

SIP defines user-agents as well as several types of server network elements [29]. Two SIP endpoints can communicate without any intervening SIP infrastructure. However, this approach is often impractical for a public service, which needs directory services to locate available nodes on the network.

3.8.1.2.1 User Agent

A SIP user agent (UA) is a logical network end-point used to create or receive SIP messages and thereby manage a SIP session. A SIP UA can perform the role of a User Agent Client (UAC), which sends SIP requests, and the User Agent Server (UAS), which receives the requests and returns a SIP response. These roles of UAC and UAS only last for the duration of a SIP transaction.

I.e. a SIP phone is an IP phone that implements SIP user agent and server functions, which provide the traditional call functions of a telephone, such as dial, answer, reject, hold/unhold, and call transfer. SIP phones may be implemented as a hardware device or as a softphone. As vendors increasingly implement SIP as a standard telephony platform, often driven by 4G efforts, the distinction between hardware-based and software-based SIP phones is being blurred and SIP elements are implemented in the basic firmware functions of many IP-capable devices. Examples are devices from Nokia and BlackBerry.

In SIP, as in HTTP, the user agent may identify itself using a message header field 'User-Agent', containing a text description of the software/hardware/product involved. The User-Agent field is sent in request messages, which means that the receiving SIP server can see this information. SIP network elements sometimes store this information and it can be useful in diagnosing SIP compatibility problems.

3.8.1.2.2 Proxy server

The proxy server is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, meaning that its job is to ensure that a request is sent to another entity closer to
the targeted user. Proxies are also useful for enforcing policies (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.

3.8.1.2.3 Registrar
A registrar is a SIP endpoint that accepts REGISTER requests and places the information it receives in those requests into a location service for the domain it handles. The location service links one or more IP addresses to the SIP URI of the registering agent. The URI uses the sip: scheme, although other protocol schemes are possible, such as tel:. More than one user agent can register at the same URI, with the result that all registered user agents receive the calls to the URI.

SIP registrars are logical elements, and are commonly co-located with SIP proxies. But it is also possible and often good for network scalability to place this location service with a redirect server.

3.8.1.2.4 Redirect server
A user agent server that generates 3xx (Redirection) responses to requests it receives, directing the client to contact an alternate set of URIs. The redirect server allows proxy servers to direct SIP session invitations to external domains.

3.8.1.2.5 Session border controller
Session border controllers serve as middle boxes between UA and SIP servers for various types of functions, including network topology hiding, and assistance in NAT traversal.

3.8.1.2.6 Gateway
Gateways can be used to interface a SIP network to other networks, such as the public switched telephone network, which use different protocols or technologies.

3.8.2 SIP Messages
There are two different types of SIP messages: requests and responses [29][32].

Note: The receiver of a request may answer with one-or-more responses. Transactions control the exchange of messages between participants. Transactions can establish long-running conversations called Dialog. Because of these transactional mechanisms, SIP can make use of un-reliable transports such as UDP.

3.8.2.1 SIP request and its methods
SIP requests and responses messages have following structure: first line, several header field lines (each line containing one name-value-pair plus optional attributes), an empty line and the body. The body is used by the embedded protocols like SDP for negotiations of the stream protocols to use in the session.
The first line of a request contains a method (also called SIP command), defining the nature of the request and a Request-URI, indicating where the request should be sent.

To list a few SIP requests methods:

- **REGISTER**: Used by a user agent to register its current location and its public SIP address to the registrar. Usually the registrar will send a response accepting this registration and tell about its validity time. [RFC 3261]
- **INVITE**: Used to establish a media session between user agents. It may be used by user agents or by servers. [RFC 3261]
- **ACK**: (Acknowledgement) is used to confirm reliable message exchange(s) after INVITE. [RFC 3261]
- **CANCEL**: Terminates a pending request or search. [RFC 3261]
- **BYE**: Terminates an existing session between user agents. [RFC 3261]
- **OPTIONS**: Used to figure out the capabilities of a device. To do so, it is not needed to set up a session. [RFC 3261]

New methods have been introduced in SIP in several RFCs. Two examples are:

- **REFER**: Used to transfer a session to a new/third participant - in telephone call context typically named “call transfer”. [RFC 3515]
- **PRACK** (Provisional Response Acknowledgement): PRACK improves network reliability by adding an acknowledgement system to the provisional responses (1xx) used also in negotiation of possible media streams. [RFC 3262]

A more complete list can be found at [33].

### 3.8.2.2 SIP response and response codes

The first line of a response has a response code. The SIP response types defined in RFC 3261 fall in one of the following categories:

- **Provisional (1xx)**: Request received and being processed.
- **Success (2xx)**: The action was successfully received, understood, and accepted.
- **Redirection (3xx)**: Further action needs to be taken (typically by sender) to complete the request.
- **Client Error (4xx)**: The request contains bad syntax or cannot be fulfilled at the server.
- **Server Error (5xx)**: The server failed to fulfill an apparently valid request.
- **Global Failure (6xx)**: The request cannot be fulfilled at any server.

More details about SIP response codes can be found at [34].

### 3.8.3 Object model mapping

As in XMPP mapping the principle is to reuse the definitions of the IEC 61850-8-1 regarding the mapping of the ACSI objects and services over the corresponding concepts of MMS and use the updates defined in the new IEC 61850-8-2: includes using ASN.1 XER (XML encoding
rule) as new concrete Message Encoding. Each IED should get an own SIP address. The user part of the SIP address could follow the recommendations of IEC 61850 naming. Resources of the IED should be addressed appropriate within the transferred messages.

3.8.4 Mapping solicited services

Due to the main purpose of SIP managing sessions between two or more communication partners and leaving the data transfer to other protocols, the SIP data model is quite different from ACSI object model and direct mapping is no preferred way. The SIP protocol design offers strong extendibility as well as for the core functionality of SIP as well as plugging-in new protocols for the data transfer/stream – so the standard can be enhanced for new applications not included in the core and existing extensions.

For the solicited services as typical request/response communication pattern there are two options:

A) Using SIP messages to transfer OS4ES data.
B) Using SIP to establish a stream between the communication partners and using it for the message-exchange between client and server.

For solution A) the SIP-standard-message “OPTIONS” would be a good choice. Another option would be to use the Instant-Messaging-SIP-Extension-Message: MESSAGE. In both cases the content-part of the HTTP-like SIP-messages - the message body / message payload – could be used by OS4ES to transfer own data in an OS4ES defined data structure, as this body is not used by SIP itself. A third option would be to specify a new SIP-extension specific for OS4ES similar to following SIP-SOAP approach that defines a new method/command and also the encoding of the payload: [30] describes a SIP-extension with a generic and extensible framework through which SIP nodes can request additional services from remote nodes using SOAP-messages inside SIP. It introduces a new SIP method/command, called SERVICE. The SERVICE method can carry a Simple Object Access Protocol (SOAP) formatted messages as its payload. The XML-Schema (XSD) the messages apply must be designed like mentioned for DPWS.

Because the transferred data-structures could be freely designed, alternative to a SOAP-XSD, the messages could be designed in the same way like XMPP mapping does for stanzas in XML: an element for AssociationContext, representing the association established between the client and the server and an element MMSpdu, which is the XML payload of the message.

For solution B) SIP could be used to establish a stream free of choice. SIPs support for negotiation between the communication partners could be used to find a common supported protocol to use. It could be also XMPP.

3.8.5 Mapping of unsolicited services

Asynchronous event handling is used in IEC 61850 for the reporting and control model. A possible communication scheme for realizing these asynchronous services could be the use of
the SIP extension Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE). It is the SIP-based suite of standards for instant messaging and presence information.

The message content can be handled the same way like for solicited services.

### 3.8.6 Information security

First, the SIP protocol belongs to the Application layer regarding the OSI reference model (short: OSI). In connection with this protocol environment, several attacks could be appear with a different impact. For instance, it is not excluded, that a Replay Attack appears. In this case valid SIP packets are stored and will be reordered to interrupt a SIP session. Eavesdropping, Man in the Middle Attacks or the theft of credentials belongs also to possible attacks [7]. A protection against those Attacks against Confidentiality, Authenticity and Integrity [8] might be a security protocols below SIP regarding the OSI reference model. IPSec is an example in this environment and belongs to OSI Level 3. This is the reason for a possible integration of IPSec and TCP /UDP without any change of details. Generally IPSec should be a preferred solution – in contrast to TLS. Because TLS is closer to the Application OSI layer and it does not provide real end-to-end security to prevent espionage and law enforcement interception.

### 3.8.7 Open source implementations

There is no known existing implementation of the above mentioned design of the combination of protocols, being fully integrated and working.

For the SIP core the U.S. National Institute of Standards and Technology (NIST), Advanced Networking Technologies Division provides a public-domain Java implementation [31] that serves as a reference implementation for the SIP standard. The implementation can work in proxy server or user agent scenarios and has been used in numerous commercial and research projects. It supports RFC 3261 in full and a number of extension RFCs including RFC 6665 (event notification) and RFC 3262 (reliable provisional responses).
3.9 Mapping comparison and comments

The Table 12 shows the summary table and mapping comparison for the all analysed middleware technologies according to the mapping criteria required for IEC 61850 communication.

- Asynchronous communication – indicates if the technology supports event-driven communication required by IEC 61850 reporting
- Data modelling – indicates if the technology supports data modelling mechanism which should be used for adapting IEC 61850 data model
- Data encoding – indicates if the technology supports Binary or XML-based data encoding
- Information Security- indicates if the technology provides application-level security mechanisms
- Open source implementation- indicates if there are freely available open source implementations which could be used for further development in OS4ES project

According the features criteria the selected technology for implementation of OS4ES communication is XMPP.

<table>
<thead>
<tr>
<th>Middleware technology / Features</th>
<th>OPC UA</th>
<th>IEC 61400-25 Annex A</th>
<th>DPWS</th>
<th>XML over WebSocket</th>
<th>XMPP</th>
<th>SIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous communication</td>
<td>+/-</td>
<td>+/-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Data modelling</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Data encoding</td>
<td>Binary /XML</td>
<td>XML</td>
<td>XML</td>
<td>XML</td>
<td>Binary</td>
<td>XML</td>
</tr>
</tbody>
</table>
| Information Security             | +      | -                   | +    | -                   | +    | -
| Open source implementations      | +      | *                   | +    | +                   | +    | +   |

Table 12: Mapping comparison

---

If SIP will be used with IPSec/VPN, the key management needs the same symmetric keys in the applied systems. A IPSec/VPN structure with SIP might be impractical and the use must be guaranteed in all used OS4ES components.
References

[1] IEC, Communication networks and systems for power utility automation - Part 7-1: Basic information and communication Communication networks and systems for power utility automation - Part 7-1: Basic communication structure - Principles and models, Int. Std. IEC 61850-7-1 ed2.0, 2011.


[18] IEC, Communication networks and systems for power utility automation - Part 7-4: Basic communication structure - Compatible logical node classes and data object classes, Int. Std. IEC 61850-7-4, ed2.0, 2010.


[31] JAIN SIP Developer Tools, public-domain Java implementation of SIP [Online] https://java.net/projects/jsip

