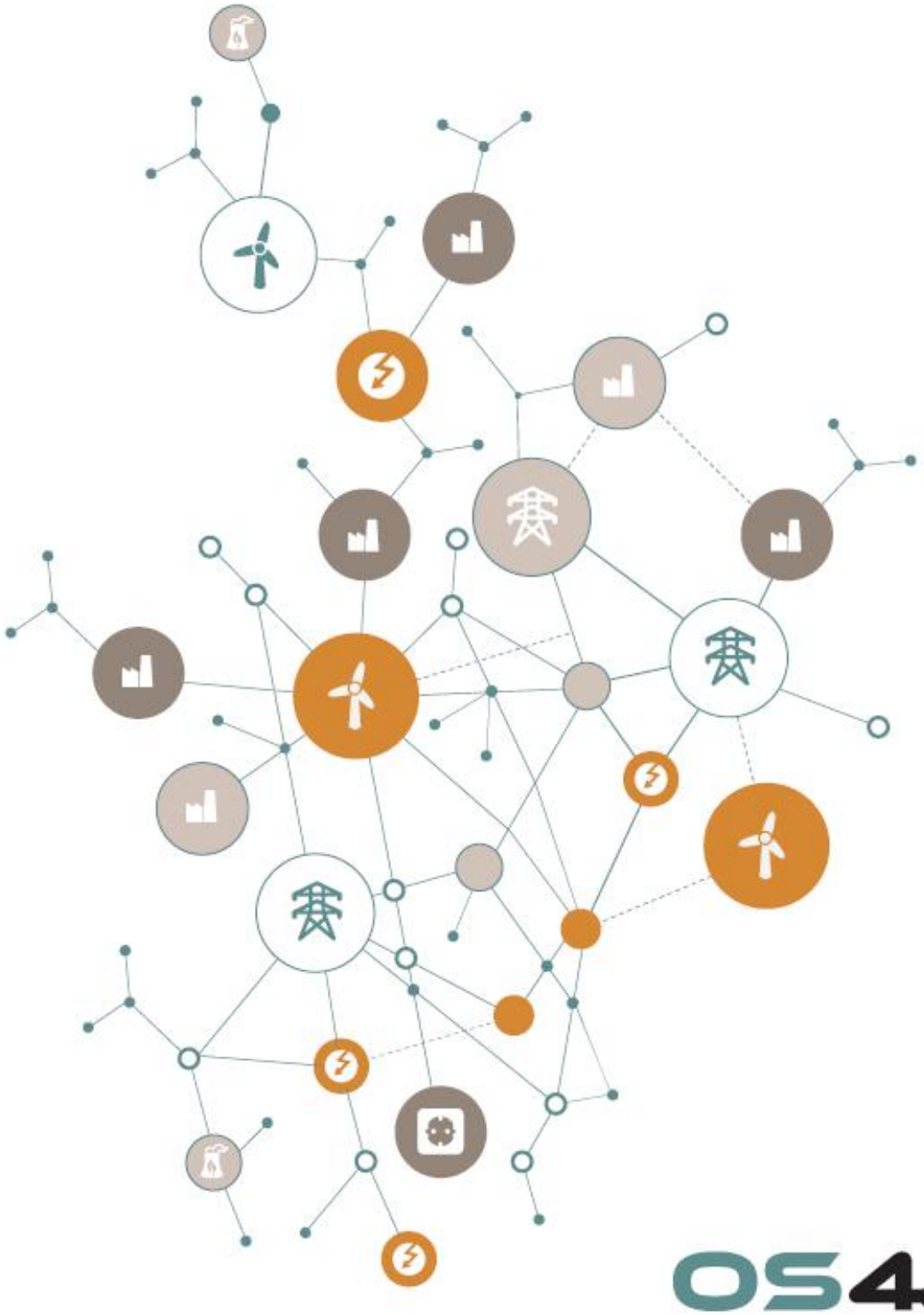
Deliverable 3.2

Communication protocol prototype and gateways development



OS4ES

D3.2 Communication protocol prototype and gateways development

Document Information

Programme	FP7-ICT-2013-11
Project acronym	OS4ES
Grant agreement number	619302
Number of the Deliverable	D3.2
WP/Task related	WP 3 – Task 3.3
Type (distribution level)	PU
Start date of project	01/07/2014
Date of delivery	01/03/2016
Status and Version	Final version 1.0
Number of pages	71 pages
Document Responsible	Stjepan Sučić – Končar
Author(s)	Stjepan Sučić – Končar
Reviewers	Markus Breuers – FGH

Revision History

Version	Date	Author/Reviewer	Notes
0.1	28.02.2015	Stjepan Sučić	Draft ToC
0.2	15.03.2016	Stjepan Sučić	First version D3.2
0.3	20.03.2016	Markus Breuers	API modifications
0.4	06.04.2016	Stjepan Sučić	Final version D3.2

Executive Summary

This deliverable provides development results related to protocol definitions found in D3.1. The deliverable includes API descriptions for IEC 61850 XMPP based communication and the Modbus gateway. This deliverable is sectioned as follows:

- Section 1 provides an introduction and table of acronyms
- Section 2 describes XMPP-based IEC 61850 communication API and describes developers environment setup
- Section 3 describes common utility functions used in the library
- Section 4 describes JSON schemas as basis for configuring modules and applications
- Section 5 describes ASN.1 compiler and XER encoder
- Section 6 describes all the communication protocol drivers used in the implementation including XMPP, COSP, ASCE, MMS and IEC 61850

Table of contents

List of Tables

1 Introduction

1.1 Scope of the document

This deliverable defines API definitions of XMPP-based 61850 client and server libraries as well as Modbus and IEC 104 client descriptions.

1.2 Notations, abbreviations and acronyms

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

ACSE	Association Control Service Element
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
COTP	Connection Oriented Transport Protocol
GCC	GNU Compiler Collection
GNU	GNU's Not Unix
HTTP	Hypertext Transfer Protocol
IP	Internet protocol
JSON	Javascript Object Notation
MMS	Manufacturing Message Specification
XER	XML Encoding Rules
XML	Extnsible Markup Language
XMPP	Extensible Messaging Presence Protocol

Table 1 : Acronyms list

2 XMPP-based IEC 61850 communication API

As a result of D3.2 an XMPP based implementation of IEC 61850 communication has been developed. It is based on utilizing Python programming language. Hat-open is collection of open source libraries used as basis for OS4ES project and is described in this document.

2.1 Dependencies

Following utilities and libraries are dependencies for building and running hat components. As such, they should be available on development machines.

Applications:

- Unix command line tools (find, bash, make, ...)
- Python >= 3.4
- GCC >= 4.8

List of all python dependencies is available in *requirements.pip.txt* file, which can be installed by command *pip install -r requirements.pip.txt*.

2.2 Developers environment setup

This section describes one of possible development machine setups. This setup is not mandatory and is given only as recommendation.

2.2.1 Windows development machine setup

Follow the next few steps to get your Windows development machine up and running:

1. Download the latest version of **MSYS2** (Minimal SYStem 2) – system that allows building native Windows applications, using bash shell, utilizing make files, etc. – from their official web page
   http://sourceforge.net/p/msys2/wiki/MSYS2%20installation/.

   Add following locations to the system path:
   - append location to \\msys64\\mingw32\\bin folder to System Variable PATH
   - append location to \\msys64\\usr\\bin folder to System Variable PATH

   If default installation directory is used, locations to be added to the system path are C:\\msys64\\mingw32\\bin and C:\\msys64\\usr\\bin

2. Execute the **msys2_shell.bat** file which can be found in the MSYS2 installation folder. In the msys2 shell use following commands:

```
3.   $ pacman -Sy
4.   $ pacman -S pacman-mirrors
```

   These commands will synchronize local package databases with the latest repositories and update pacman download mirrors.

After that, by using **pacman** (PACkage MANager) install all required packages (defined in **requirements.pacman.win.txt**). To do so, use your msys2 shell - position yourself in the folder where *requirements.pacman.win.txt*file is located and execute the following command:

```
$ pacman -S $(< requirements.pacman.win.txt)
```

5. Download the **Python** programming language version 3.5.x from their official web page *https://www.python.org/*. Add Python to the system path:
   - prepend the System Variable PATH with the path to Python's Scripts folder
   - prepend the System Variable PATH with the path to python.exe
6. Configure python to use *gcc* as default compiler creating file PYTHONPATH\Lib\distutils\distutils.cfg with content:

```
[build]
compiler=gcc
```

9. Use **pip** – Python's tool for installing Python packages – to install all required packages (python libraries defined in **requirements.pip.txt**)
   To do so, use your Windows command line - position yourself in the folder where *requirements.pip.txt* file is located and execute the following command:

```
$ pip install -r requirements.pip.txt
```

10. Download and install TeX Live from *http://www.tug.org/texlive/*. Add binary directory (which contains pdflatex.exe) to System Variable PATH.

Note

After completing all the steps above your System Variable PATH should look like python_paths; old_paths; mingw_paths; msys_paths; texlive_paths; where old_paths is the System Variable PATH defined as before the start of this setup.

2.2.2 Arch Linux development machine setup

1. Install pacman packages:

```
$ pacman -Sy --needed --noconfirm $(< requirements.pacman.linux.txt)
```

2. Install pip packages:

```
$ pip install -r requirements.pip.txt
```

2.3 Documentation

Additional developer's documentation can be built by running doit docs_build. After build process finishes, documentation is available in *build/docs* folder.

3 Utility functions

3.1 Python implementation

`hat.util.`**`first`**(*xs*, *fn*)

Find first element in collection that satisfies predicat

Parameters:	**xs** (*Iterable[Any]*) – collection
	fn (*Callable[[Any],bool]*) – predicate
Returns:	Any

`hat.util.`**`uint_to_bebytes`**(*x*)

Convert unsigned integer to bigendian bytes

Parameters:	**x** (*int*) – unsigned integer
Returns:	bytes

`hat.util.`**`bebytes_to_uint`**(*b*)

Convert bigendian bytes to unsigned integer

Parameters:	**b** (*bytes*) – bigendian bytes
Returns:	int

`hat.util.`**`is_asn1_oid_eq`**(*x*, *y*)

Check if two ASN.1 object identifiers are equal

Parameters:	**x** (*hat.asn1.encoder.ObjectIdentifier*) – object identifier
	y (*hat.asn1.encoder.ObjectIdentifier*) – object identifier
Returns:	bool

`hat.util.`**`logging_local_setup`**(*config*)

Initializes logging facilities

Configuration parameters are defined by `hat://logging.yaml#`

Parameters:	**config** (*Dict[str,Any]*) – Configuration parameters

class `hat.util.`**`Timestamp`**

Bases: **`hat.util.Timestamp`**

Implemented as named tuple.

Objects of this class support comparison operators.

s

int

seconds since 1970-01-01

us

int

microseconds added to timestamp defined by seconds

to_float()

Convert timestamp to floating number of seconds since 1970-01-01

Returns:	timestamp
Return type:	float

static **from_float**(*ts*)

Create new timestamp from floating number of seconds since 1970-01-01

Parameters:	**ts** (*float*) – seconds since 1970-01-01
Returns:	timestamp
Return type:	hat.util.Timestamp

to_datetime()

Convert timestamp to datetime

Returns:	datetime
Return type:	datetime.datetime

static **from_datetime**(*dt*)

Create new timestamp from datetime

It is assumed that provided datetime represents utc time.

Parameters:	**dt** (*datetime.datetime*) – datetime
Returns:	timestamp
Return type:	hat.util.Timestamp

static **now**()

Create new timestamp representing current time

Returns:	timestamp
Return type:	hat.util.Timestamp

exception hat.util.**QueueClosedError**

Bases: **Exception**

Error signaling closed queue

class hat.util.**Queue**(**vargs*, ***kwargs*)

Bases: **asyncio.queues.Queue**

Extension of asyncio Queue with close method

is_closed

Is closed

Returns: bool

close()

Close queue

get()

See **asyncio.Queue.get()**

Raises: **QueueClosedError**

put(*item*)¶

See **asyncio.Queue.put()**

Raises: **QueueClosedError**

put_nowait(*item*)

See **asyncio.Queue.put_nowait()**

Raises: **QueueClosedError**

4 JSON Schemas

Preferred way of providing configuration data is by data structures which can be mapped to JSON data. JSON Schemas are used for definition of such data. All schemas used in Hat should have identifiers starting with *hat://*.

4.1 Python implementation

`hat.json_validator.`**`validate`**(*data*, *schema_id*)

Validate data with JSON schema

Parameters:	**data** – validated data **schema_id** (*str*) – JSON schema identificator
Raises:	**`Exception`** – validation fails

4.2 JSON Schema definitions

logging.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://logging.yaml#"
title: Logging
description: Logging configuration
type: object
required:
    - file
    - default_level
    - levels
    - file_size
    - file_count
properties:
    file:
        title: Log file
        description: >
            Path to the file where log messages will be written
        type: string
        minLength: 1
    default_level:
        title: Default logging level
        description: Minimum level of messages to log
        enum:
            - NOTSET
            - CRITICAL
            - ERROR
            - WARNING
            - INFO
            - DEBUG
        default: INFO
    levels:
        title: Logging levels for specific loggers
        description: Keys represent logger names
```

```
        type: object
        patternProperties:
            "(.)+":
                title: Logging level
                enum:
                    - NOTSET
                    - CRITICAL
                    - ERROR
                    - WARNING
                    - INFO
                    - DEBUG
    file_size:
        title: File size
        description: >
            Maximum log file size in KB (creates a new log file after the
            current file size reaches this)
        type: number
        default: 10240
        minimum: 0
    file_count:
        title: File count
        description: >
            Maximum number of log files (when this number of log files is
            reached each new log file will overwrite the oldest log file)
        type: number
        default: 5
        minimum: 0
        multipleOf: 1
...
```

modbus61850.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://modbus61850.yaml#"
title: Modbus61850
description: Modbus61850 configuration
type: object
required:
    - server
    - client
properties:
    server:
        title: IEC61850 server
        anyOf:
            - allOf:
                - "$ref": "hat://drivers/iec61850/server.yaml#"
                - "$ref": "hat://drivers/mms/server.yaml#"
                - "$ref": "hat://drivers/copp/server.yaml#"
                - "$ref": "hat://drivers/cosp/server.yaml#"
                - "$ref": "hat://drivers/cotp/server.yaml#"
            - allOf:
                - "$ref": "hat://drivers/iec61850/server.yaml#"
                - "$ref": "hat://drivers/mms/server.yaml#"
                - "$ref": "hat://drivers/copp/server.yaml#"
                - "$ref": "hat://drivers/xmppsp/server.yaml#"
                - "$ref": "hat://drivers/xmpp.yaml#"
    client:
        title: Modbus master
        type: object
        required:
```

```
            - address
            - device_id
            - poll_interval
        properties:
            address:
                type: string
            device_id:
                type: integer
            poll_interval:
                descriptions: polling interval in seconds
                type: number
    translations:
        title: Translations
        type: array
        items:
            title: Translation
            type: object
            required:
                - direction
                - iec61850_logical_device
                - iec61850_address
                - iec61850_type
                - modbus_type
                - modbus_address
                - modbus_quantity
            properties:
                direction:
                    enum:
                        - modbus_to_iec61850
                        - iec61850_to_modbus
                iec61850_logical_device:
                    type: string
                iec61850_address:
                    type: array
                    items:
                      type: string
                iec61850_type:
                    enum:
                        - bool
                        - int8
                        - int32
                        - float
                modbus_type:
                    enum:
                        - COIL
                        - DISCRETE_INPUT
                        - HOLDING_REGISTER
                        - INPUT_REGISTER
                modbus_address:
                    type: integer
                modbus_quantity:
                    type: integer
...
```

drivers\xmpp.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/xmpp.yaml#"
title: XMPP client configuration
```

```
type: object
required:
required:
    - xmpp
properties:
    xmpp:
        title: XMPP client configuration
        type: object
        required:
            - host
            - port
            - jid
            - password
        properties:
            host:
                title: Host name
                type: string
            port:
                title: TCP port
                type: integer
                default: 5222
            jid:
                title: Jabber Identifier
                type: string
            password:
                title: Password
                type: string
...
```

drivers\copp\client.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/copp/client.yaml#"
title: COPP client configuration
type: object
required:
    - copp
properties:
    copp:
        title: COPP client
        type: object
        required:
            - src_psel
            - dst_psel
            - session_protocol
        properties:
            src_psel:
                title: Source PSEL
                type: integer
            dst_psel:
                title: Destination PSEL
                type: integer
            session_protocol:
                title: Session protocol
                enum:
                    - cosp
                    - xmppsp
...
```

drivers\copp\server.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/copp/server.yaml#"
title: COPP server configuration
type: object
required:
    - copp
properties:
    copp:
        title: COPP server
        type: object
        required:
            - src_psel
            - session_protocol
        properties:
            src_psel:
                title: Source PSEL
                type: integer
            session_protocol:
                title: Session protocol
                enum:
                    - cosp
                    - xmppsp
...
```

drivers\cosp\client.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/cosp/client.yaml#"
title: COSP client configuration
type: object
required:
    - cosp
properties:
    cosp:
        title: COSP client
        type: object
        required:
            - src_ssel
            - dst_ssel
        properties:
            src_ssel:
                title: Source SSEL
                type: integer
            dst_ssel:
                title: Destination SSEL
                type: integer
...
```

drivers\cosp\server.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/cosp/server.yaml#"
title: COSP server configuration
type: object
required:
```

```
    - cosp
properties:
    cosp:
        title: COSP server
        type: object
        required:
            - src_ssel
        properties:
            src_ssel:
                title: Source SSEL
                type: integer
...
```

drivers\cotp\client.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/cotp/client.yaml#"
title: COTP client configuration
type: object
required:
    - tcp
    - cotp
properties:
    tcp:
        title: TCP configuration
        type: object
        required:
            - host
            - port
        properties:
            host:
                title: Host name
                description: Remote hostname
                type: string
            port:
                title: TCP port
                type: integer
                default: 102
    cotp:
        title: COTP client
        type: object
        required:
            - src_tsel
            - dst_tsel
        properties:
            src_tsel:
                title: Source TSEL
                type: integer
            dst_tsel:
                title: Destination TSEL
                type: integer
...
```

drivers\iec61850\client.yaml

```
---
```

```
$schema: http://json-schema.org/schema#
id: hat://drivers/iec61850/client.yaml#
title: IEC61850 client configuration
type: object
required:
    - iec61850
properties:
    iec61850:
        type: object
        required:
            - status_timeout
            - separator
        properties:
            status_timeout:
                title: Check status timeout
                type: number
            separator:
                title: Flat element address segment separator
                type: string
...
```

drivers\iec61850\server.yaml

```
---
$schema: http://json-schema.org/schema#
id: hat://drivers/iec61850/server.yaml#
title: IEC61850 server configuration
type: object
required:
    - iec61850
properties:
    iec61850:
        type: object
        required:
            - separator
            - logical_device_separator
            - ied_name
            - logical_devices
            - references
        properties:
            separator:
                type: string
            logical_device_separator:
                type: string
            ied_name:
                type: string
            logical_devices:
                title: All logical devices
                type: array
                items:
                    title: Single logical device
                    type: object
                    required:
                        - name
                        - elements
                        - datasets
                    properties:
                        name:
                            type: string
                        elements:
```

```
                            title: All elements in single logical device
                            type: array
                            items:
                                "$ref":
"hat://drivers/iec61850/server.yaml#definitions/element"
                        datasets:
                            title: All datasets in single logical device
                            type: array
                            items:
                                "$ref":
"hat://drivers/iec61850/server.yaml#definitions/dataset"
            references:
                type: object
                description: |
                    Keys are reference names and values are referenced
types
                patternProperties:
                    "(.)+":
                        "$ref":
"hat://drivers/iec61850/server.yaml#definitions/element_type"
definitions:
    element:
        type: object
        required:
            - name
            - type
        properties:
            name:
                title: Element name
                type: string
            type:
                oneOf:
                    - type: string
                      description: |
                        This element is reference to definition defined in
                        references field
                    -                                                   "$ref":
"hat://drivers/iec61850/server.yaml#definitions/element_type"
            value:
                "$ref":
"hat://drivers/iec61850/server.yaml#definitions/data_value"
    element_type:
        oneOf:
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/array_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/bcd_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/bit_string_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/boolean_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/floating_point_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/integer_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/mms_string_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/octet_string_td"
            -                                                           "$ref":
"hat://drivers/iec61850/server.yaml#definitions/structure_td"
```

```
            -                                                                   "$ref":
"hat://drivers/iec61850/server.yaml#definitions/unsigned_td"
            -                                                                   "$ref":
"hat://drivers/iec61850/server.yaml#definitions/utc_time_td"
            -                                                                   "$ref":
"hat://drivers/iec61850/server.yaml#definitions/bin_time_td"
            -                                                                   "$ref":
"hat://drivers/iec61850/server.yaml#definitions/visible_string_td"
    array_td:
        type: object
        properties:
            tname:
                enum:
                    - array
            length:
                type: integer
            typeof:
                "$ref":
"hat://drivers/iec61850/server.yaml#definitions/element_type"
    bcd_td:
        type: object
        properties:
            tname:
                enum:
                    - bcd
            xyz:
                type: integer
    bit_string_td:
        type: object
        properties:
            tname:
                enum:
                    - bit_string
            length:
                type: integer
    boolean_td:
        type: object
        properties:
            tname:
                enum:
                    - bool
    floating_point_td:
        type: object
        properties:
            tname:
                enum:
                    - float
            format_width:
                type: integer
            exponent_width:
                type: integer
    integer_td:
        type: object
        properties:
            tname:
                enum:
                    - int
            size:
                type: integer
    mms_string_td:
        type: object
```

```
        properties:
            tname:
                enum:
                    - mms_string
            xyz:
                type: integer
    octet_string_td:
        type: object
        properties:
            tname:
                enum:
                    - octet_string
            length:
                type: integer
    structure_td:
        type: object
        properties:
            tname:
                enum:
                    - structure
            elements:
                type: array
                items:
                    "$ref":
"hat://drivers/iec61850/server.yaml#definitions/element"
    unsigned_td:
        type: object
        properties:
            tname:
                enum:
                    - uint
            size:
                type: integer
    utc_time_td:
        type: object
        properties:
            tname:
                enum:
                    - utc_time
    bin_time_td:
        type: object
        properties:
            tname:
                enum:
                    - bin_time
            size:
                description: If true, bin_time includes date.
                type: boolean
    visible_string_td:
        type: object
        properties:
            tname:
                enum:
                    - visible_string
            length:
                type: integer
    data_value:
        anyOf:
            - description: Used for array and structure
              type: array
              items:
```

```
                "$ref":
"hat://drivers/iec61850/server.yaml#definitions/data_value"
            - description: Used for int, uint an bcd
              type: integer
            - description: Used for bit_string
              type: array
              items:
                  type: boolean
            - description: Used for bool
              type: boolean
            - description: Used for float
              type: number
            - description: Used for mms_string and visible_string
              type: string
            - description: Used for octet_string
              type: array
              items:
                  type: integer
            - description: Used for utc_time
              type: string
              # pattern: "^[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-
9]{2}.[0-9]3Z ([0-9]|[a-f]){2}$"
    dataset:
        type: object
        required:
            - name
            - items
        properties:
            name:
                title: Full name
                description: |
                    Full name is defined as <LN>$<DS> (is separator is $)
            items:
                title: All items in single dataset
                type: array
                items:
                    title: Single dataset item
                    type: array
                    description: |
                        elements of this array are LD, LN, FC, DO, ...
                    items:
                        type: string
...
```

drivers\mms\client.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/xmppsp/client.yaml#"
title: XMPPSP client configuration
type: object
required:
    - xmppsp
properties:
    xmppsp:
        title: XMPPSP client
        type: object
        required:
            - encoding
            - server_id
```

```
            - connect_timeout
        properties:
            encoding:
                enum:
                    - base64
                    - xml
            server_id:
                type: string
            connect_timeout:
                description: connect timeout in seconds
                type: number
...
```

drivers\mms\server.yaml

```
--
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/xmppsp/server.yaml#"
title: XMPPSP server configuration
type: object
required:
    - xmppsp
properties:
    xmppsp:
        title: XMPPSP server
        type: object
        required:
            - encoding
        properties:
            encoding:
                enum:
                    - base64
                    - xml
...
```

rivers\xmppsp\client.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/xmppsp/client.yaml#"
title: XMPPSP client configuration
type: object
required:
    - xmppsp
properties:
    xmppsp:
        title: XMPPSP client
        type: object
        required:
            - encoding
            - server_id
            - connect_timeout
        properties:
            encoding:
                enum:
                    - base64
                    - xml
            server_id:
                type: string
            connect_timeout:
                description: connect timeout in seconds
                type: number
...
```

drivers\xmppsp\server.yaml

```
---
"$schema": "http://json-schema.org/schema#"
id: "hat://drivers/xmppsp/server.yaml#"
title: XMPPSP server configuration
type: object
```

```
required:
    - xmppsp
properties:
    xmppsp:
        title: XMPPSP server
        type: object
        required:
            - encoding
        properties:
            encoding:
                enum:
                    - base64
                    - xml
...
```

5 ASN.1

5.1 Python implementation

Generic ASN.1 encoder interface

Any instance of ASN.1 type can be represented by instance of Python type as following:

Table 2 ASN.1 and Python type mapping

ASN.1 type	Python type
Boolean	bool
Integer	int
BitString	List[bool]
OctetString	bytes
Null	None
ObjectIdentifier	hat.asn1.encoder.ObjectIdentifier
Utf8String	str
VisibleString	str
External	Dict[str,Any]
Choice	Tuple[str,Any]
Set	Dict[str,Any]
SetOf	Iterable[Any]
Sequence	Dict[str,Any]
SequenceOf	List[Any]
ABSTRACT-SYNTAX.&Type	Entity

For Choice, Set and Sequence, *str* represents field name.

External is represented by dictionary containing keys ‘data’, ‘direct_ref’ and ‘indirect_ref’. Key ‘data’ is mandatory and is associated with external data (Union[Entity,bytes,List[bool]]). Keys ‘direct_ref’ and ‘indirect_ref’ are optional and represent direct reference (List[int]) and indirect reference (int).

Type ObjectIdentifier is defined as *typing.List[typing.Union[int, typing.Tuple[str, int]]]*.

class `hat.asn1.encoder.`**`Entity`**

Bases: **`object`**

ABC generic entity

All encoding specific entities are subclasses of this class.

class `hat.asn1.encoder.`**`Encoder`**

Bases: **`hat.asn1.encoder.Encoder`**

Base class for translator between encoding entities and Python data structures

`repository`

hat.asn1.definition.Repository

repository

`syntax_name`

Syntax name

Returns: ObjectIdentifier

`parser`

Entity parser

This parser

Returns: Callable[[bytes],hat.asn1.encoder.Entity]

`encode`(*module_name*, *type_name*, *data*)

Encode data to encoding entity

Parameters: **module_name** (*str*) – module name
type_name (*str*) – user defined type name
data (*Any*) – data

Returns: encoding entity specific for concrete encoder implementation

Return type: hat.asn1.encoder.Entity

Raises: **`Exception`**

`decode`(*module_name*, *type_name*, *entity*)

Decode data from encoding entity

Parameters: **module_name** (*str*) – module name

	type_name (*str*) – user defined type name **entity** (*hat.asn1.encoderEntity*) – encoding entity specific for concrete encoder implementation
Returns:	Any
Raises:	**Exception**

5.2 XER Encoder

ASN.1 XML Encoding Rules

class hat.asn1.xer.**Entity**(*element*)

Bases: **hat.asn1.encoder.Entity**

Entity representing data in XML format

Parameters:	**element** (*xml.etree.ElementTree.Element*) – XML tree root element

__bytes__()

Encode entity to bytes representation

Returns:	encoded entity
Return type:	bytes
Raises:	**Exception**

static **parse**(*data*)

Decode entity

Parameters:	**entity_bytes** (*bytes*) – encoded entity bytes
Returns:	entity
Return type:	hat.asn1.xer.Entity
Raises:	**Exception**

class hat.asn1.xer.**XerEncoder**

Bases: **hat.asn1.encoder.Encoder**

Encoder implementation for XER

encode(*module_name*, *type_name*, *data*)

Encode data to Entity

Parameters:	**module_name** (*str*) – module name **type_name** (*str*) – user defined type name **data** (*Any*) – data
Returns:	hat.asn1.xer.Entity
Raises:	**Exception**

decode(*module_name*, *type_name*, *entity*)

Decode data from Entity

Parameters:	**module_name** (*str*) – module name **type_name** (*str*) – user defined type name **entity** (*hat.asn1.xer.Entity*) – entity
Returns:	Any
Raises:	**Exception**

6 Communication protocol drivers

6.1 XMPP client

XMPP client is base don using SleekXMPP librarires and is developed as asyncio wrapper.

`hat.drivers.xmpp.`**`connect`**(*conf*, *loop=None*)

COROUTINE - Connect to XMPP server

Parameters:	**conf** (*dict*) – configuration specified by 'hat://drivers/xmpp.yaml#'
	loop (*asyncio.BaseEventLoop*) – event loop
Returns:	Optional[Connection]

class `hat.drivers.xmpp.`**`Connection`**(*xmpp*)

Bases: **`object`**

XMPP connection

User should not call Connection's constructor. For creating new connection see **`hat.drivers.xmpp.connect()`**

Parameters:	**xmpp** (*hat.drivers.xmpp._XMPPWrapper*) – instance of xmpp wrapper

`read`()
COROUTINE - Read

Returns:	message; None on error or if connection is closed
Return type:	Optional[Msg]

`write`(*msg*)
Write

Write message's from attribute is ignored and replaced with current connection's user id.

Parameters:	**msg** (*Msg*) – message

`close`()
Close the connection

class `hat.drivers.xmpp.`**`Msg`**

Bases: hat.drivers.xmpp.Msg

XMPP message

from_id

str

sender's id

to_id

str

receiver's id

msg_type

str

message type identification

xml_data¶

str

xml string containing message payload

6.1.1 XMPP based Service Protocol

XMPP based Service Protocol

hat.drivers.xmppsp.**connect**(*conf*, *user_data*)

COROUTINE - Connect to XMPPSP server

Parameters:	**conf** (*dict*) – configuration specified by 'hat://drivers/xmppsp/client.yaml#' and 'hat://drivers/xmpp.yaml#'
	user_data (*Optional[bytes]*) – connect request user data
Returns:	Optional[Connection]

hat.drivers.xmppsp.**listen**(*conf*, *validate_connection_cb*, *new_connection_cb*)

COROUTINE - Create XMPPSP listening server

Parameters:	**conf** (*dict*) – configuration specified by 'hat://drivers/xmppsp/server.yaml#' and 'hat://drivers/xmpp.yaml#'
	validate_connection_cb (*Callable[[bytes],ValidateConnectionResult]*) – callback function called on new incomming connection request prior to creating connection object - function receives incomming request's user data and returns validation result
	new_connection_cb (*Callable[[Connection],None]*) – new connection callback
Returns:	Server

class hat.drivers.xmppsp.**ValidateConnectionResult**

Bases: hat.drivers.xmppsp.ValidateConnectionResult

Result of incomming connection user validation

accepted
bool
connection is acceptable

res_user_data
Optional[bytes]
response user data

class hat.drivers.xmppsp.**Server**
Bases: **object**
XMPPSP listening server
User should not call Server's constructor. For creating new server see **hat.drivers.xmppsp.listen()**

close()
Close listening server
Closing server does not close active incomming connections created by this server.

class hat.drivers.xmppsp.**Connection**(*conf*, *xmpp_conn*, *conn_id*, *remote_id*, *conn_req_user_data*, *conn_res_user_data*)
Bases: **object**
XMPPSP connection
User should not call Connection's constructor. For creating new connection see **hat.drivers.xmppsp.connect()**

conn_req_user_data
Connect request's user data
Returns: bytes

conn_res_user_data
Connect response's user data
Returns: bytes

read()
COROUTINE - Read
Returns: data; None on error or if connection is closed
Return type: Optional[bytes]

write(*data*)
Write
Parameters: **data** (*bytes*) – data

close(*user_data=None*)
Close connection

Parameters: **user_data** (*Optional[bytes]*) – finish message user data

6.1.2 Connection Oriented Presentation Protocol

hat.drivers.copp.**connect**(*conf*, *encoder*, *syntax_names*, *user_data*)
COROUTINE - Connect to COPP server

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/copp/client.yaml#' and 'hat://drivers/cosp/client.yaml#' and 'hat://drivers/cotp/client.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/copp/client.yaml#' and 'hat://drivers/xmppsp/client.yaml#' and 'hat://drivers/xmpp.yaml#'. **encoder** (*hat.asn1.encoder.Encoder*) – ASN.1 encoder **syntax_names** (*List[hat.asn1.encoder.ObjectIdentifier]*) – list of ASN.1 ObjectIdentifiers representing syntax names **user_data** (*Optional[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]*) – connect request user data
Returns:	Optional[Connection]

`hat.drivers.copp.`**`listen`**(*conf*, *encoder*, *validate_connection_cb*, *new_connection_cb*)

COROUTINE - Create COPP listening server

Server return by this method is object with method `close()` which can be called in case server should be closed (by closing server, all active connections remain open and only listening socket is closed).

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/copp/server.yaml#' and 'hat://drivers/cosp/server.yaml#' and 'hat://drivers/cotp/server.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/copp/server.yaml#' and 'hat://drivers/xmppsp/server.yaml#' and 'hat://drivers/xmpp.yaml#'. **encoder** (*hat.asn1.encoder.Encoder*) – ASN.1 encoder **validate_connection_cb** (*Callable[[Dict[int,hat.asn1.encoder.ObjectIdentifier],Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]],ValidateConnectionResult]*) – callback function called on new incomming connection request prior to creating connection object - function receives incomming request's syntax names (with syntax ids) and user data and returns validation result **new_connection_cb** (*Callable[[Connection],None]*) – new connection callback
Returns:	Server

class `hat.drivers.copp.`**`ValidateConnectionResult`**

Bases: **`hat.drivers.copp.ValidateConnectionResult`**

Result of incomming connection user validation

accepted

bool

connection is acceptable

res_user_data

Optional[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]

response user data

class hat.drivers.copp.**Connection**(*cosp_conn*, *encoder*, *cp_ppdu*, *cpa_ppdu*)

Bases: **object**

COPP connection

User should not call Connection's constructor. For creating new connection see **hat.drivers.copp.connect()**

conn_req_user_data

Connect request's user data

Returns: Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]

conn_res_user_data

Connect response's user data

Returns: Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]

get_syntax_name(*syntax_id*)

Get syntax name by syntax id

Parameters: **syntax_id** (*int*) – syntax id

Returns: hat.asn1.encoder.ObjectIdentifier

get_syntax_id(*syntax_name*)

Get syntax id by syntax name

Parameters: **syntax_name** (*hat.asn1.encoder.ObjectIdentifier*) – syntax name

Returns: int

read()

COROUTINE - Read

Returns: data; None on error or if connection is closed

Return type: Optiona[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]

write(*data*)

Write

Parameters: **data** (*Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]*) – data

close(*user_data=None*)

Close connection

Parameters:	**user_data** (*Optional[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]*) – closing message user data

6.1.3 Association Control Service Element

`hat.drivers.acse.`**`connect`**(*conf*, *encoder*, *app_context_name*, *syntax_names*, *user_data*)

COROUTINE - Connect to ACSE server

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/copp/client.yaml#' and 'hat://drivers/cosp/client.yaml#' and 'hat://drivers/cotp/client.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/copp/client.yaml#' and 'hat://drivers/xmppsp/client.yaml#' and 'hat://drivers/xmpp.yaml#'. **encoder** (*hat.asn1.encoder.Encoder*) – ASN.1 encoder **app_context_name** (*hat.asn1.encoder.ObjectIdentifier*) – application context name **syntax_names** (*List[hat.asn1.encoder.ObjectIdentifier]*) – list of ASN.1 ObjectIdentifiers representing syntax names **user_data** (*Optional[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]*) – connect request user data
Returns:	Optional[Connection]

`hat.drivers.acse.`**`listen`**(*conf*, *encoder*, *validate_connection_cb*, *new_connection_cb*)

COROUTINE - Create ACSE listening server

Server return by this method is object with method `close()` which can be called in case server should be closed (by closing server, all active connections remain open and only listening socket is closed).

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/copp/server.yaml#' and 'hat://drivers/cosp/server.yaml#' and 'hat://drivers/cotp/server.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/copp/server.yaml#' and

'hat://drivers/xmppsp/server.yaml#' and 'hat://drivers/xmpp.yaml#'.

encoder (*hat.asn1.encoder.Encoder*) – ASN.1 encoder

validate_connection_cb (*Callable[[Dict[int,hat.asn1.encoder.ObjectIdentifier],Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]],ValidateConnectionResult]*) – callback function called on new incomming connection request prior to creating connection object - function receives incomming request's syntax names (with syntax ids) and user data and returns validation result

new_connection_cb (*Callable[[Connection],None]*) – new connection callback

Returns: Server

class hat.drivers.acse.**ValidateConnectionResult**

Bases: **hat.drivers.acse.ValidateConnectionResult**

Result of incomming connection user validation

accepted

bool

connection is acceptable

res_user_data

Optional[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]]

response user data

class hat.drivers.acse.**Connection**(*copp_conn*, *encoder*, *aarq_apdu*, *aare_apdu*)

Bases: **object**

6.1.3.1 ACSE connection

User should not call Connection's constructor. For creating new connection see **hat.drivers.acse.connect()**

conn_req_user_data

Connect request's user data

Returns: Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]

conn_res_user_data

Connect response's user data

Returns: Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]

read()

COROUTINE - Read

Returns: data; None on error or if connection is closed

Return type: Optiona[Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.En

tity]]

write(*data*)

Write

Parameters: **data** (*Tuple[hat.asn1.encoder.ObjectIdentifier,hat.asn1.encoder.Entity]*) – data

close()¶

Close connection

6.1.4 Manufacturing Message Specification

6.1.4.1 Client

hat.drivers.mms.client.**connect**(*conf*)

COROUTINE - Connect to MMS server

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/mms/client.yaml#' and 'hat://drivers/copp/client.yaml#' and 'hat://drivers/cosp/client.yaml#' and 'hat://drivers/cotp/client.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/mms/client.yaml#' and 'hat://drivers/copp/client.yaml#' and 'hat://drivers/xmppsp/client.yaml#' and 'hat://drivers/xmpp.yaml#'.
Returns:	Optional[Connection]

class hat.drivers.mms.client.**Connection**(*conf*, *encoder*, *acse_conn*, *initiate_req*, *initiate_res*)

Bases: **object**

MMS client connection

User should not call Connection's constructor. For creating new connection see **hat.drivers.mms.client.connect()**

is_connected

Is connected

Returns: bool

get_confirmed(*request*)

COROUTINE - Get response for confirmed request

Parameters: **request** (*RequestBase*) – request

Returns: Optional[ResponseBase]

get_unconfirmed()

COROUTINE - Get unconfirmed message

Returns:	Optional[UnconfirmedBase]

close()

Close connection

6.1.4.2 Server

Manufacturing Message Specification Server

hat.drivers.mms.server.**listen**(*conf*, *new_connection_cb*, *close_cb*, *confirmed_cb*)

COROUTINE - Create MMS listening server

Server return by this method is object with method `close()` which can be called in case server should be closed (by closing server, all active connections remain open and only listening socket is closed).

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/mms/server.yaml#' and 'hat://drivers/copp/server.yaml#' and 'hat://drivers/cosp/server.yaml#' and 'hat://drivers/cotp/server.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/mms/server.yaml#' and 'hat://drivers/copp/server.yaml#' and 'hat://drivers/xmppsp/server.yaml#' and 'hat://drivers/xmpp.yaml#'. **new_connection_cb** (*Optional[Callable[[Connection],None]]*) – new connection callback **close_cb** (*Optional[Callable[[Connection],None]]*) – connection closed callback **confirmed_cb** (*Optional[Callable[[Connection,hat.drivers.mms.common.RequestBase],hat.drivers.mms.common.ResponseBase]]*) – confirmed request callback (can be plain function or coroutine)
Returns:	Server

class hat.drivers.mms.server.**Connection**(*conf*, *encoder*, *acse_conn*, *close_cb*, *confirmed_cb*)

Bases: **object**

MMS server connection

User should not call Connection's constructor. For creating new connection see **hat.drivers.mms.server.listen()**

is_connected

Is connected

Returns: bool

send_unconfirmed(*unconfirmed*)

COROUTINE - Send unconfirmed message

Parameters: **unconfirmed** (*UnconfirmedBase*) – unconfirmed message

close()

Close connection

6.1.4.3 Common classes

class hat.drivers.mms.common.data.**DataAccessError**

Bases: **enum.Enum**

An enumeration.

class hat.drivers.mms.common.data.**DataBase**

Bases: **object**

Base class for all data

class hat.drivers.mms.common.data.**ArrayData**

Bases: **hat.drivers.mms.common.data.ArrayData**, **hat.drivers.mms.common.data.DataBase**

Array data

elements

List[DataBase]

elements

class hat.drivers.mms.common.data.**BcdData**

Bases: **hat.drivers.mms.common.data.BcdData**, **hat.drivers.mms.common.data.DataBase**

BCD data

value

int

value

class hat.drivers.mms.common.data.**BinaryTimeData**

Bases: **hat.drivers.mms.common.data.BinaryTimeData**, **hat.drivers.mms.common.data.DataBase**

BinaryTime data

value

bytes

value

to_datetime()

Convert to datetime

Returns: datetime

static **from_datetime**(*utc_dt*)

Create new BinaryTimeData from datetime

Parameters:	**utc_dt** (*datetime*) – utc datetime
Returns:	BinaryTimeData

class hat.drivers.mms.common.data.**BitStringData**

Bases: **hat.drivers.mms.common.data.BitStringData**, **hat.drivers.mms.common.data.DataBase**

BitString data

value

List[bool]

Value

class hat.drivers.mms.common.data.**BooleanData**

Bases: **hat.drivers.mms.common.data.BooleanData**, **hat.drivers.mms.common.data.DataBase**

Boolean data

value

bool

value

class hat.drivers.mms.common.data.**BooleanArrayData**

Bases: **hat.drivers.mms.common.data.BooleanArrayData**, **hat.drivers.mms.common.data.DataBase**

BooleanArray data

value

List[bool]

Value

class hat.drivers.mms.common.data.**FloatingPointData**

Bases: **hat.drivers.mms.common.data.FloatingPointData**, **hat.drivers.mms.common.data.DataBase**

FloatingPoint data

value

bytes

value

static **from_float**(*x*)

Create FloatingPointData from floating value

Parameters:	**x** (*float*) – floating point value

Returns: FloatingPointData

class hat.drivers.mms.common.data.**GeneralizedTimeData**

Bases: **hat.drivers.mms.common.data.GeneralizedTimeData**, **hat.drivers.mms.common.data.DataBase**

GeneralizedTime data

value

value

class hat.drivers.mms.common.data.**IntegerData**

Bases: **hat.drivers.mms.common.data.IntegerData**, **hat.drivers.mms.common.data.DataBase**

Integer data

value

int

value

class hat.drivers.mms.common.data.**MmsStringData**

Bases: **hat.drivers.mms.common.data.MmsStringData**, **hat.drivers.mms.common.data.DataBase**

MmsString data

value

str

value

class hat.drivers.mms.common.data.**ObjIdData**

Bases: **hat.drivers.mms.common.data.ObjIdData**, **hat.drivers.mms.common.data.DataBase**

ObjId data

value

hat.asn1.encoder.ObjectIdentifier

value

class hat.drivers.mms.common.data.**OctetStringData**

Bases: **hat.drivers.mms.common.data.OctetStringData**, **hat.drivers.mms.common.data.DataBase**

OctetString data

value

bytes

value

class hat.drivers.mms.common.data.**StructureData**

Bases: **hat.drivers.mms.common.data.StructureData**, **hat.drivers.mms.common.data.DataBase**

Structure data

elements

List[DataBase]

elements

class hat.drivers.mms.common.data.**UnsignedData**

Bases: **hat.drivers.mms.common.data.UnsignedData**, **hat.drivers.mms.common.data.DataBase**

Unsigned data

value

int

value

class hat.drivers.mms.common.data.**UtcTimeData**

Bases: **hat.drivers.mms.common.data.UtcTimeData**, **hat.drivers.mms.common.data.DataBase**

UtcTime data

value

bytes

value

static **from_datetime**(*utc_dt*, *quality*)

Create UtcTimeData from datetime and quality

Parameters:	**utc_dt** (*datetime*) – utc datetime **quality** (*int*) – quality
Returns:	UtcTimeData

class hat.drivers.mms.common.data.**VisibleStringData**

Bases: **hat.drivers.mms.common.data.VisibleStringData**, **hat.drivers.mms.common.data.DataBase**

VisibleString data

value

str

value

class hat.drivers.mms.common.request.**RequestBase**

Bases: **object**

Base class for all confirmed requests

encode_pdu(*invoke_id*)

Encode PDU

Parameters:	**invoke_id** (*int*) – invoke id
Returns:	MMS PDU
Return type:	Any

static **decode_pdu**(*pdu*)

Decode PDU

Parameters:	**service** (*Any*) – MMS PDU
Returns:	invoke id and request
Return type:	Tuple[int,RequestBase]

class hat.drivers.mms.common.request.**StatusRequest**

Bases: **hat.drivers.mms.common.request.StatusRequest**, **hat.drivers.mms.common.request.RequestBase**

Status request

class hat.drivers.mms.common.request.**GetNameListRequest**

Bases: **hat.drivers.mms.common.request.GetNameListRequest**, **hat.drivers.mms.common.request.RequestBase**

GetNameList request

object_class

ObjectClass

object class

object_scope

ObjectScopeBase

object scope

continue_after

Optional[str]

continue after

class hat.drivers.mms.common.request.**GetVariableAccessAttributesRequest**

Bases: **hat.drivers.mms.common.request.GetVariableAccessAttributesRequest**, **hat.drivers.mms.common.request.RequestBase**

GetVariableAccessAttributes request

value

Union[Address,ObjectNameBase]

address or object name

class hat.drivers.mms.common.request.**GetNamedVariableListAttributesRequest**

Bases: **hat.drivers.mms.common.request.GetNamedVariableListAttributesRequest**, **hat.drivers.mms.common.request.RequestBase**

GetNamedVariableListAttributes request

value

ObjectNameBase

object name

class hat.drivers.mms.common.request.**ReadRequest**

Bases: **hat.drivers.mms.common.request.ReadRequest**, **hat.drivers.mms.common.request.RequestBase**

Read request

value

Union[List[VariableSpecificationBase],ObjectNameBase]

value

class hat.drivers.mms.common.request.**WriteRequest**

Bases: **hat.drivers.mms.common.request.WriteRequest**, **hat.drivers.mms.common.request.RequestBase**

Write request

specification

Union[List[VariableSpecificationBase],ObjectNameBase]

variable access specification

data

List[DataBase]

list of data

class hat.drivers.mms.common.request.**DefineNamedVariableListRequest**

Bases: **hat.drivers.mms.common.request.DefineNamedVariableListRequest**, **hat.drivers.mms.common.request.RequestBase**

DefineNamedVariableList request

variableListName

ObjectNameBase

variable list name

listOfVariable
List[VariableSpecificationBase]
variable access specifications

class hat.drivers.mms.common.request.**DeleteNamedVariableListRequest**

Bases: **hat.drivers.mms.common.request.DeleteNamedVariableListRequest**, **hat.drivers.mms.common.request.RequestBase**

DeleteNamedVariableList request
listOfVariableListName
List[ObjectNameBase]
list of variable list names

class hat.drivers.mms.common.response.**ResponseBase**

Bases: **object**
Base class for all confirmed responses
encode_pdu(*invoke_id*)
Encode PDU

Parameters:	**invoke_id** (*int*) – invoke id
Returns:	MMS PDU
Return type:	Any

static **decode_pdu**(*pdu*)¶
Decode PDU

Parameters:	**service** (*Any*) – MMS PDU
Returns:	invoke id and response
Return type:	Tuple[int,ResponseBase]

class hat.drivers.mms.common.response.**ErrorResponse**
Bases: **hat.drivers.mms.common.response.ErrorResponse**, **hat.drivers.mms.common.response.ResponseBase**

Error response

class hat.drivers.mms.common.response.**StatusResponse**
Bases: **hat.drivers.mms.common.response.StatusResponse**, **hat.drivers.mms.common.response.ResponseBase**

Status request

logical
int
vmd logical status
physical
int
vmd physical status

class hat.drivers.mms.common.response.**GetNameListResponse**
Bases: **hat.drivers.mms.common.response.GetNameListResponse**, **hat.drivers.mms.common.response.ResponseBase**

GetNameList response
identifiers
List[str]
identifiers
more_follows
bool
more follows

class hat.drivers.mms.common.response.**GetVariableAccessAttributesResponse**
Bases: **hat.drivers.mms.common.response.GetVariableAccessAttributesResponse**, **hat.drivers.mms.common.response.ResponseBase**

GetVariableAccessAttributes response

mms_deletable
bool

MMS deletable

type_description
TypeDescriptionBase

type description

class hat.drivers.mms.common.response.**GetNamedVariableListAttributesResponse**
Bases: **hat.drivers.mms.common.response.GetNamedVariableListAttributesResponse**, **hat.drivers.mms.common.response.ResponseBase**

GetNamedVariableListAttributes response

mms_deletable
bool

MMS deletable

specification
List[VariableSpecificationBase]

variable specification

class hat.drivers.mms.common.response.**ReadResponse**

Bases: **hat.drivers.mms.common.response.ReadResponse**, **hat.drivers.mms.common.response.ResponseBase**

Read response

results
List[Union[DataAccessError,DataBase]]

results

class hat.drivers.mms.common.response.**WriteResponse**

Bases: **hat.drivers.mms.common.response.WriteResponse**, **hat.drivers.mms.common.response.ResponseBase**

Write response
results
List[Optional[DataAccessError]]
results (None represents success)

class hat.drivers.mms.common.response.**DefineNamedVariableListResponse**

Bases: **hat.drivers.mms.common.response.DefineNamedVariableListResponse**, **hat.drivers.mms.common.response.ResponseBase**

DefineNamedVariableList response

class hat.drivers.mms.common.response.**DeleteNamedVariableListResponse**

Bases: **hat.drivers.mms.common.response.DeleteNamedVariableListResponse**, **hat.drivers.mms.common.response.ResponseBase**

DeleteNamedVariableList response
numberMatched
int
number of matched named variable lists
numberDeleted
int
number of deleted named variable lists

class hat.drivers.mms.common.types.**TypeDescriptionBase**

Bases: **object**

Base class for all type descriptions

class hat.drivers.mms.common.types.**ArrayTypeDescription**

Bases: **hat.drivers.mms.common.types.ArrayTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

Array type description

number_of_elements

int

nuber of elements

element_type

Union[TypeDescriptionBase,ObjectNameBase]

element type

class hat.drivers.mms.common.types.**BcdTypeDescription**

Bases: **hat.drivers.mms.common.types.BcdTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

BCD type description

class hat.drivers.mms.common.types.**BinaryTimeTypeDescription**

Bases: **hat.drivers.mms.common.types.BinaryTimeTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

BinaryTime type description

class hat.drivers.mms.common.types.**BitStringTypeDescription**

Bases: **hat.drivers.mms.common.types.BitStringTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

BitString type description

int

class hat.drivers.mms.common.types.**BooleanTypeDescription**

Bases: **hat.drivers.mms.common.types.BooleanTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

Boolean type description

class hat.drivers.mms.common.types.**FloatingPointTypeDescription**

Bases: **hat.drivers.mms.common.types.FloatingPointTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

FloatingPoint type description

format_width

int

format width

exponent_width

int

exponent width

class hat.drivers.mms.common.types.**GeneralizedTimeTypeDescription**

Bases: **hat.drivers.mms.common.types.GeneralizedTimeTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

GeneralizedTime type description

class hat.drivers.mms.common.types.**IntegerTypeDescription**

Bases: **hat.drivers.mms.common.types.IntegerTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

Integer type description

int

class hat.drivers.mms.common.types.**MmsStringTypeDescription**

Bases: **hat.drivers.mms.common.types.MmsStringTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

MmsString type description

int

class hat.drivers.mms.common.types.**ObjIdTypeDescription**

Bases: **hat.drivers.mms.common.types.ObjIdTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

ObjId type description

class hat.drivers.mms.common.types.**OctetStringTypeDescription**

Bases: **hat.drivers.mms.common.types.OctetStringTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

OctetString type description

int

class hat.drivers.mms.common.types.**StructureTypeDescription**

Bases: **hat.drivers.mms.common.types.StructureTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

Structure type description

components

List[Component]

components

class **Component**

Bases: **hat.drivers.mms.common.types.Component**

Single structure's component

component_name

Optional[str]

name

component_type

Union[TypeDescriptionBase,ObjectNameBase]

type

class hat.drivers.mms.common.types.**UnsignedTypeDescription**

Bases: **hat.drivers.mms.common.types.UnsignedTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

Unsigned type description

int

class hat.drivers.mms.common.types.**UtcTimeTypeDescription**

Bases: **hat.drivers.mms.common.types.UtcTimeTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

UtcTime type description

class hat.drivers.mms.common.types.**VisibleStringTypeDescription**

Bases: **hat.drivers.mms.common.types.VisibleStringTypeDescription**, **hat.drivers.mms.common.types.TypeDescriptionBase**

VisibleString type description

int

class hat.drivers.mms.common.types.**ObjectClass**

Bases: **enum.Enum**

An enumeration.

class hat.drivers.mms.common.types.**ObjectScopeBase**

Bases: **object**

Base class for all object scopes

class hat.drivers.mms.common.types.**AaSpecificObjectScope**

Bases: **hat.drivers.mms.common.types.AaSpecificObjectScope**, **hat.drivers.mms.common.types.ObjectScopeBase**

aaSpecific object scope

class hat.drivers.mms.common.types.**DomainSpecificObjectScope**

Bases: **hat.drivers.mms.common.types.DomainSpecificObjectScope**, **hat.drivers.mms.common.types.ObjectScopeBase**

domainSpecific object scope

identifier
str

identifier

class hat.drivers.mms.common.types.**VmdSpecificObjectScope**

Bases: **hat.drivers.mms.common.types.VmdSpecificObjectScope**, **hat.drivers.mms.common.types.ObjectScopeBase**

vmdSpecific object scope

class hat.drivers.mms.common.types.**Address**

Bases: **hat.drivers.mms.common.types.Address**

MMS Address

value
Union[int,str,bytes]

address

class hat.drivers.mms.common.types.**ObjectNameBase**

Bases: **object**

Base class for all object names

class hat.drivers.mms.common.types.**AaSpecificObjectName**

Bases: **hat.drivers.mms.common.types.AaSpecificObjectName**, **hat.drivers.mms.common.types.ObjectNameBase**

AaSpecific object name

identifier

str

identifier

class hat.drivers.mms.common.types.**DomainSpecificObjectName**

Bases: **hat.drivers.mms.common.types.DomainSpecificObjectName**, **hat.drivers.mms.common.types.ObjectNameBase**

DomainSpecificObjectName object name

domain_id

str

domain id

item_id

str

item id

class hat.drivers.mms.common.types.**VmdSpecificObjectName**

Bases: **hat.drivers.mms.common.types.VmdSpecificObjectName**, **hat.drivers.mms.common.types.ObjectNameBase**

VmdSpecific object name

identifier

str

identifier

class hat.drivers.mms.common.types.**VariableSpecificationBase**

Bases: **object**

Base class for all variable specifications

class hat.drivers.mms.common.types.**AddresVariableSpecification**

Bases: **hat.drivers.mms.common.types.AddresVariableSpecification**, **hat.drivers.mms.common.types.VariableSpecificationBase**

Address variable specification

address

Address

address

class hat.drivers.mms.common.types.**InvalidatedVariableSpecification**

Bases: **hat.drivers.mms.common.types.InvalidatedVariableSpecification**, **hat.drivers.mms.common.types.VariableSpecificationBase**

Invalidated variable specification

class hat.drivers.mms.common.types.**NameVariableSpecification**

Bases: **hat.drivers.mms.common.types.NameVariableSpecification**, **hat.drivers.mms.common.types.VariableSpecificationBase**

Name variable specification

name

ObjectNameBase

name

class hat.drivers.mms.common.types.**ScatteredAccessDescriptionVariableSpecification**

Bases: **hat.drivers.mms.common.types.ScatteredAccessDescriptionVariableSpecification**, **hat.drivers.mms.common.types.VariableSpecificationBase**

ScatteredAccessDescription variable specification

specifications

List[VariableSpecificationBase]

specifications

class hat.drivers.mms.common.types.**VariableDescriptionVariableSpecification**

Bases: **hat.drivers.mms.common.types.VariableDescriptionVariableSpecification**, **hat.drivers.mms.common.types.VariableSpecificationBase**

VariableDescription variable specification

address

Address

address

type_specification

Union[TypeDescriptionBase,ObjectNameBase]

type description or type name

class hat.drivers.mms.common.unconfirmed.**UnconfirmedBase**

Bases: **object**

Base class for all unconfirmed message

encode_pdu()

Encode PDU

Returns: MMS PDU

Return type: Any

static **decode_pdu**(*pdu*)

Decode PDU

Parameters: **service** (*Any*) – MMS PDU

Returns: UnconfirmedBase

class hat.drivers.mms.common.unconfirmed.**EventNotificationUnconfirmed**

Bases: **hat.drivers.mms.common.unconfirmed.EventNotificationUnconfirmed**, **hat.drivers.mms.common.unconfirmed.UnconfirmedBase**

Event notification

class hat.drivers.mms.common.unconfirmed.**InformationReportUnconfirmed**

Bases: **hat.drivers.mms.common.unconfirmed.InformationReportUnconfirmed**, **hat.drivers.mms.common.unconfirmed.UnconfirmedBase**

Information report

specification

Union[List[VariableSpecificationBase],ObjectNameBase]

variable access specification

data

List[Union[DataAccessError,DataBase]]

data

class hat.drivers.mms.common.unconfirmed.**UnsolicitedStatusUnconfirmed**

Bases: **hat.drivers.mms.common.unconfirmed.UnsolicitedStatusUnconfirmed**, **hat.drivers.mms.common.unconfirmed.UnconfirmedBase**

Unsolicited status

logical

int

vmd logical status

physical

int

vmd physical status

6.2 IEC 61850

6.2.1 IEC 61850 Client

`hat.drivers.iec61850.client.`**`connect`**(*conf*)

COROUTINE - Connect to IEC 61850 server

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/iec61850/client.yaml#' and 'hat://drivers/copp/client.yaml#' and 'hat://drivers/cosp/client.yaml#' and 'hat://drivers/cotp/client.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/iec61850/client.yaml#' and 'hat://drivers/mms/client.yaml#' and 'hat://drivers/copp/client.yaml#' and 'hat://drivers/xmppsp/client.yaml#' and 'hat://drivers/xmpp.yaml#'.
Returns:	Optional[Connection]

class `hat.drivers.iec61850.client.`**`Connection`**(*conf*, *mms_conn*)

Bases: **`object`**

IEC 61850 client connection

User should not call Connection's constructor. For creating new connection see **`hat.drivers.iec61850.client.connect()`**

`is_connected`

Is connected

Returns: bool

`browser`

Connection's browser

Returns: Browser

`reader`

Connection's reader

Returns: Reader

`writer`

Connection's writer

Returns: Writer

`ds_manager`

Connection's data set manager

Returns: DsManager

`close`()

Close connection

class hat.drivers.iec61850.client.**ReportData**

Bases: **hat.drivers.iec61850.client.ReportData**

Single report data entry

result

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

value

reference

Optional[str]

data reference

reason_codes

Optional[List[bool]]

reason codes

class hat.drivers.iec61850.client.**Report**

Bases: **hat.drivers.iec61850.client.Report**

rpt_id

str

report id

seq_num

Optional[int]

sequence number

time_of_entry

Optional[datetime]

report timestamp

data_set

Optional[str]

dataset name

buf_ovfl

Optional[bool]

buffer overflow flag

entry_id

Optional[bytes]

entry id

conf_rev

Optional[int]

configuration revision

sub_seq_num

Optional[int]

subsequence number

more_segments_follow

Optional[int]

more segments follow

data

List[ReportData]

data entries

class hat.drivers.iec61850.client.**Browser**(*conn*)

Bases: **object**

OptFlds are mandatory in a report header but hat.drivers.iec61850.Report class doesn't include it because the data from the report is already parsed. All other report header attributes are Optinal['type'] meaning they will provide either attribute value for the given 'type' or None if they are omitted in OptFlds.

Also, OptFlds value can be reconstructed from the Report object as following:

def get_opt_flds(report):

Get OptFlds from report.

Returns:

List[bool]

```
optFlds = [False for i in range(10)]
optFlds[1] = report.seq_num != None
optFlds[2] = report.time_of_entry != None
optFlds[3] = report.data[0].reason_codes != None
optFlds[4] = report.data_set != None
optFlds[5] = report.data[0].reference != None
optFlds[6] = report.buf_ovfl != None
optFlds[7] = report.entry_id != None
optFlds[8] = report.conf_rev != None
optFlds[9] = report.sub_seq_num != None
return optFlds
```

Connection’s interface for performing browse actions

get_server_directory()

COROUTINE - Get server directory
Returns list of all logical devices registered on server.

Returns:	logical devices or None on error
Return type:	Optional[List[str]]

get_logical_device_directory(*logical_device*)

COROUTINE - Get logical device directory
Returns list of all logical nodes registered on logical device.

Returns:	logical nodes or None on error
Return type:	Optional[List[str]]

get_logical_node_directory(*logical_device*, *logical_node*, *acsi_class*)¶

COROUTINE - Get logical node directory

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **acsi_class** (*AcsiClass*) – acsi class
Returns:	if acsi_class is DATA_SET then returned value is list of data set names (List[str]) List[LogicalNodeEntry]: if acsi_class is not DATA_SET then returned value is list of logical node entries None: on error
Return type:	List[str]

get_data_directory(*logical_device*, *logical_node*, *data_object*, *functional_constraint*)

COROUTINE - Get data directory

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – data object **functional_constraint** (*str*) – functional constraint
Returns:	data object elements
Return type:	Optional[List[str]]

get_data_set_directory(*logical_device*, *logical_node*, *data_set*)

COROUTINE - Get data set directory

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_set** (*str*) – data set name
Returns:	list of data set entries (each entry is represented as ordered list [<LD>, <LN>, <FC>, <DO>, ...])
Return type:	Optional[List[List[str]]]

get_data_definition(*logical_device*, *logical_node*, *data_object*, *data_object_element*, *functional_constraint*)

COROUTINE - Get data directory

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – data object **data_object_element** (*str*) – data object element (data object or data attribute) **functional_constraint** (*str*) – functional constraint
Returns:	type description
Return type:	Optional[hat.drivers.mms.common.TypeDescriptionBase]

class hat.drivers.iec61850.client.**Reader**(*conn*)

Bases: **object**

Connection's interface for performing read actions

select(*logical_device*, *logical_node*, *data_object*)

COROUTINE - Command select before operate without value.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – command data object
Returns:	data value
Return type:	Optional[DataBase]

Note: Return type should be Optional[VisibleStringData] but it's server dependant.

get_urcb_values(*logical_device*, *logical_node*, *urcb*, *urcb_da*)

COROUTINE - Get unbuffered report control block values

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **urcb** (*str*) – unbuffered report control block **urcb_da** (*str*) – data attribute
Returns:	data attribute's value
Return type:	Optional[hat.drivers.mms.common.DataBase]

get_brcb_values(*logical_device*, *logical_node*, *brcb*, *brcb_da*)

COROUTINE - Get buffered report control block values

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **brcb** (*str*) – buffered report control block **brcb_da** (*str*) – data attribute
Returns:	data attribute's value
Return type:	Optional[hat.drivers.mms.common.DataBase]

get_edit_sg_value(*logical_device*, *logical_node*, *functional_constraint*, *path*)

COROUTINE - Read value(s) from Setting Group.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **functional_constraint** (*str*) – functional constraint ('SG' or 'SE') **path** (*List[str]*) – single item path ([<DO>, ..., <DA>, ...])
Returns:	data value
Return type:	Optional[DataBase]

get_sgcb_values(*logical_device*)

COROUTINE - Read SGCB attribute values.

Parameters:	**logical_device** (*str*) – logical device
Returns:	data value
Return type:	Optional[DataBase]

Return example should be as follows.

Example

```
>>> StructureData([
```

```
    UnsignedData,        # NumOfSG
    UnsignedData,        # ActSG
    UnsignedData,        # EditSG
    BooleanData,         # CnfEdit
    UtcTimeData,         # LActTm
    IntegerData          # ResvTms (optional)
])
```

get_data_values(*logical_device*, *logical_node*, *functional_constraint*, *path*)

COROUTINE - Get data values

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **functional_constraint** (*str*) – functional constraint **path** (*List[str]*) – single item path ([<DO>, ..., <DA>, ...])
Returns:	data value
Return type:	Optional[hat.drivers.mms.common.DataBase]

get_report()

COROUTINE - Get next report

Returns:	Optional[Report]

class hat.drivers.iec61850.client.**Writer**(*conn*)

Bases: **object**

Connection's interface for performing write actions

set_urcb_values(*logical_device*, *logical_node*, *urcb*, *urcb_da*, *data*)

COROUTINE - Set unbuffered report control block values

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **urcb** (*str*) – unbuffered report control block **urcb_da** (*str*) – data attribute **data** (*DataBase*) – data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

set_brcb_values(*logical_device*, *logical_node*, *brcb*, *brcb_da*, *data*)

COROUTINE - Set buffered report control block values

Parameters:	**logical_device** (*str*) – logical device

	logical_node (*str*) – logical node **brcb** (*str*) – buffered report control block **brcb_da** (*str*) – data attribute **data** (*DataBase*) – data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

select_with_value(*logical_device*, *logical_node*, *data_object*, *data*)

COROUTINE - Command select before operate with value.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – command data object **data** (*DataBase*) – SBOw block data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

Parameter **data** should conform to SBOw block of command (look at example).

Example

```
>>> data = StructureData([
    BooleanData(True),                                  # ctlVal
    UtcTimeData.from_datetime(datetime.datetime(
        1970, 1, 1, tzinfo=datetime.timezone.utc), 0),  # OperTm
    StructureData([                                     # origin
        IntegerData(3),                                 # orCat
        OctetStringData(b'.\x6f\x74\x45')               # orIdent
    ]),
    UnsignedData(0),                                    # ctlNum
    UtcTimeData.from_datetime(
        datetime.datetime.utcnow(), 0),                 # T
    BooleanData(False),                                 # Test
    BitStringData([False, False])                       # Check
])
```

Note: ctlVal can differ by its type

operate(*logical_device*, *logical_node*, *data_object*, *data*)

COROUTINE - Command operate.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – command data object **data** (*DataBase*) – operate block data

	value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

Parameter **data** should conform to Oper block of command (look at example).

Example

```
>>> data = StructureData([
    BooleanData(True),                                  # ctlVal
    UtcTimeData.from_datetime(datetime.datetime(
        1970, 1, 1, tzinfo=datetime.timezone.utc), 0),  # OperTm
    StructureData([                                     # origin
        IntegerData(3),                                 # orCat
        OctetStringData(b'.\x6f\x74\x45')               # orIdent
    ]),
    UnsignedData(0),                                    # ctlNum
    UtcTimeData.from_datetime(
        datetime.datetime.utcnow(), 0),                 # T
    BooleanData(False),                                 # Test
    BitStringData([False, False])                       # Check
])
```

Note: ctlVal can differ by its type

cancel(*logical_device*, *logical_node*, *data_object*, *data*)

COROUTINE - Command cancel.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **data_object** (*str*) – command data object **data** (*DataBase*) – cancel block data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

Parameter **data** should conform to Cancel block of command (look at example).

Example

```
>>> data = StructureData([
    BooleanData(True),                                  # ctlVal
    UtcTimeData.from_datetime(datetime.datetime(
        1970, 1, 1, tzinfo=datetime.timezone.utc), 0),  # OperTm
```

```
    StructureData([                                         # origin
        IntegerData(3),                                     # orCat
        OctetStringData(b'.\x6f\x74\x45')                   # orIdent
    ]),
    UnsignedData(0),                                        # ctlNum
    UtcTimeData.from_datetime(
        datetime.datetime.utcnow(), 0),                     # T
    BooleanData(False)                                      # Test
])
```

Note: ctlVal can differ by its type

select_active_sg(*logical_device*, *data*)

COROUTINE - Select active setting group.

Parameters:	**logical_device** (*str*) – logical device **data** (*UnsignedData*) – setting group number as unsigned integer size 8
Returns:	write result
Return type:	Optional[SetDataValuesResult]

select_edit_sg(*logical_device*, *data*)

COROUTINE - Select editing setting group.

Parameters:	**logical_device** (*str*) – logical device **data** (*UnsignedData*) – setting group number as unsigned integer size 8
Returns:	write result
Return type:	Optional[SetDataValuesResult]

set_edit_sg_value(*logical_device*, *logical_node*, *path*, *data*)

COROUTINE - Write value(s) to the editing Setting Group.

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **path** (*List[str]*) – single item path ([<DO>, ..., <DA>, ...]) **data** (*DataBase*) – data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

confirm_edit_sg_values(*logical_device*, *data*)

COROUTINE - Confirm editing of the Setting Group.

Parameters:	**logical_device** (*str*) – logical device **data** (*BooleanData*) – confirm edit with True

Returns:	write result
Return type:	Optional[SetDataValuesResult]

set_data_values(*logical_device*, *logical_node*, *functional_constraint*, *path*, *data*)

COROUTINE - Set data values

Parameters:	**logical_device** (*str*) – logical device **logical_node** (*str*) – logical node **functional_constraint** (*str*) – functional constraint **path** (*List[str]*) – single item path ([<DO>, ..., <DA>, ...]) **data** (*DataBase*) – data value
Returns:	write result
Return type:	Optional[SetDataValuesResult]

class hat.drivers.iec61850.client.**DsManager**(*conn*)

Bases: **object**

Connection's interface for performing data set managing actions

create_data_set(*data_set_ld*, *data_set_ln*, *data_set_name*, *ds_members*)

COROUTINE - Create data set

Parameters:	**data_set_ld** (*str*) – data set logical device **data_set_ln** (*str*) – data set logical node **data_set_name** (*str*) – data set name **ds_members** (*List[List[str]]*) – list of data set members where each member is represented as ordered list ([<LD>, <LN>, <FC>, <DO>, ...])
Returns:	create data set result
Return type:	Optional[Response]

delete_data_set(*data_set_references*)

COROUTINE - Delete data sets

Parameters:	**ds_references** (*List[List[str]]*) – list of data set references represented as ordered list ([<LD>, <LN>, <ds_name>])
Returns:	delete data set result
Return type:	Optional[Response]

class hat.drivers.iec61850.client.**SetDataValuesResult**

Bases: **enum.Enum**

An enumeration.

class hat.drivers.iec61850.client.**Response**

Bases: **enum.Enum**

An enumeration.

class hat.drivers.iec61850.client.**AcsiClass**

Bases: **enum.Enum**

An enumeration.

class hat.drivers.iec61850.client.**LogicalNodeEntry**

Bases: **hat.drivers.iec61850.client.LogicalNodeEntry**

Logical node entry

data_object

str

data object

functional_constraints

List[str]

functional constraints

6.2.2 IEC 61850 Server

hat.drivers.iec61850.server.**create_server**(*conf*, *data_change_cb*, *command_cb*)

COROUTINE - Create IEC 61850 server

Parameters:	**conf** (*dict*) – In case of OSI stack communication, configuration is specified by 'hat://drivers/iec61850/server.yaml#' and 'hat://drivers/copp/server.yaml#' and 'hat://drivers/cosp/server.yaml#' and 'hat://drivers/cotp/server.yaml#'. In case of XMPP based communication, configuration is specified by 'hat://drivers/iec61850/server.yaml#' and 'hat://drivers/mms/server.yaml#' and 'hat://drivers/copp/server.yaml#' and 'hat://drivers/xmppsp/server.yaml#' and 'hat://drivers/xmpp.yaml#'. **data_change_cb** (*Optional[Callable[[str,List[str],hat.drivers.mms.common.DataBase],None]]*) – data change callback; arguments are logical device, address and data **command_cb** (*Optional[Callable[[str,List[str],hat.drivers.mms.common.DataBase],bool]]*) – command verification callback (can be regular function or coroutine); arguments are logical device, address and data; if it returns True, command is starts execution process
Returns:	Server

Raises:	**Exception**

class hat.drivers.iec61850.server.**Server**(*conf*, *data_change_cb*, *command_cb*)

Bases: **object**

IEC 61850 server

User should not call Server's constructor. For creating new server see **hat.drivers.iec61850.server.create_server()**

get_data(*logical_device*, *address*)

Get data

Parameters:	**logical_device** (*str*) – local device **address** (*List[str]*) – item's address
Returns:	Optional[hat.drivers.mms.common.DataBase]

set_data(*logical_device*, *address*, *data*)

Set data

Parameters:	**logical_device** (*str*) – local device **address** (*List[str]*) – item's address **data** (*hat.drivers.mms.common.DataBase*) – data

close()

Close server and all active connections

6.2.3 Modbus

Modbus high-level communication interface

class hat.drivers.modbus.client.**DataType**

Bases: **enum.Enum**

Data type

COIL

hat.drivers.modbus.client.DataType

DISCRETE_INPUT

hat.drivers.modbus.client.DataType

HOLDING_REGISTER

hat.drivers.modbus.client.DataType

INPUT_REGISTER

hat.drivers.modbus.client.DataType

QUEUE

hat.drivers.modbus.client.DataType

exception hat.drivers.modbus.client.**ModbusError**(*exception_type=None*)

Bases: **Exception**

Modbus error

Objects of this class are created internally on error

hat.drivers.modbus.client.**connect**(*address*, *device_id*, *response_timeout=5*)

COROUTINE - Connect to remote device using TCP modbus

Address can be formated as:

- tcp://<host>:<port>
- rtu://<serial_port>?baudrate=<baudrate>
- ascii://<serial_port>?baudrate=<baudrate>

Parameters:	**address** (*str*) – remote server's address **device_id** (*int*) – device id (unit id or slave address) **response_timeout** (*float*) – response timeout
Returns:	client
Return type:	hat.drivers.modbus.client.Client

class hat.drivers.modbus.client.**Client**(*transport*, *address*, *with_locking*)

Bases: **object**

User should not create instance of Client directly - use **hat.drivers.modbus.client.connect()** instead

close()

Close connection

write(*data_type*, *start_address*, *values*)

COROUTINE - Write data to modbus device

Parameters:	**data_type** (*hat.drivers.modbus.client.DataType*) – data type **start_address** (*int*) – starting modbus data address **values** (*Union[List[bool],List[int]]*) – values
Returns:	is write successful
Return type:	bool

Throws:

hat.drivers.modbus.client.ModbusError:

read(*data_type*, *start_address*, *quantity=1*)

COROUTINE - Read data from modbus device

Parameters:	**data_type** (*hat.drivers.modbus.client.DataType*) – data type **start_address** (*int*) – starting modbus data address **quantity** (*int*) – number of data values (for queue this parameter is ignored)
Returns:	values
Return type:	Union[List[bool],List[int]]

Throws:

hat.drivers.modbus.client.ModbusError: