sos-access Documentation

Release 0.0.5

Palmlund Wahlgren Innovative Technology AB

Aug 26, 2021

Contents:

CHAPTER 1

Installation

sos-access requires python version > 3.6

To install use pip:

```
pip install sos-access
```

CHAPTER 2

Create a Client

A client represents one transmitter. If you want to send alarms from several transmitters you need to create several client instances.

The SOS Alarm Access v4 requires you to set up two receivers when sending alarm. For development and testing this is not really necessary. We have the use_single_receiver to tell the client that it is ok that only one receiver was supplied.

For creating clients that sends the alarms encrypted to the alarm receiver set use_tls to True

```
client = SOSAccessClient(
    transmitter_code='IK00001',
    transmitter_type='SV001',
    authentication='012345678912345',
    receiver_address=('alarm.example.com',1234),
    receiver_id='ALARM-OPER',
    use_single_receiver=True,
    use_tls=True
    )
```

CHAPTER 3

Send an Alarm

It is easy to send an alarm. Just use the .send_alarm() function. The minimal you need to supply is an event_code. You need to agree on an event code schema with your alarm operator to make actions in case of an alarm.

```
# Simple invocation
response = client.send_alarm(event_code='AL')


# All options
response = client.send_alarm(
    event_code='AL',
    transmitter_time=datetime.datetime.now(),
    reference='my_first_alarm',
    transmitter_area='12345',
    section='1',
    section_text='my_section',
    detector='1',
    detector_text='my_detector',
    additional_info={'key': 'test'},
    position=None
    )
```

Note: additional_info supports receiving a string, dict or an iterable (list, set, tuple). If dict it will be generated as a string with all keys and values on different rows. If an iterable it will be converted into a string with all items on different rows.

Note: As of now there is no good handling of position data. You will need to supply the correctly formatted position yourself.

CHAPTER 4

Send a ping

Ping is sent via the .ping() method.

```
response = client.ping(reference='my-ping')  # reference will only appear in logs.
```

CHAPTER 5

Change password of transmitter

It is possible to change the password of the transmitter. You need to be aware that the client does not persist this password anywhere. If you change the password you need to collect the new password returned from the receiver server and store it for later use.

```
new_auth_response = client.request_new_auth()

my_save_pass_func(new_auth_response.new_authentication)
```

Note: The new password only starts working after the first new transmission using it. Until then you can use the old password.

CHAPTER 6

Retries

The client implements a retry functionality between the primary and secondary alarm receiver.

In the specification of the SOS Access v4 protocol there is nothing hindering the client from keep alternately retrying each server for ever. But this is not practical. The standard value of retry for the client is 3 times on each receiver.

If you need to change this then subclass the client and change MAX_RETRY

```python
class ManyRetryClient(SOSAccessClient):
    MAX_RETRY = 100
```

CHAPTER 7

SOS Access v4 Protocol

SOS Access V4 protocol is intended for transmission of alarm signals between alarm transmitters and receivers.

Functions for monitoring the transmission link are included.

It is an XML based protocol and only printable 8-bits characters from ISO8859-1 are allowed.

The protocol consists of the following message types:

7.1 Alarm Requests

<alarmrequest> Alarm, reset or information message

<alarmresponse> Response message for <alarmrequests>

7.2 Change password

<requestnewauthentication> Request new password

<requestnewauthenticationresponse> Response message for <requestnewauthentication> messages, containing the new password

7.3 Monitored Links

<pingrequest> Hearbeat message from the alarm transmitter

<pingresponse> Response message for <pingrequest>

7.4 Client Implementation

In the sos-access client the messages are modeled by the following classes

<alarmrequest> -> *AlarmRequest*

<alarmresponse> -> *AlarmResponse*

<requestnewauthentication> -> *NewAuthRequest*

<requestnewauthenticationresponse> -> *NewAuthResponse*

<pingrequest> -> *PingRequest*

<pingresponse> -> *PingResponse*

CHAPTER 8

Alarm Transmission

<alarmrequest> is the telegram that is used when an alarm is sent to the alarm receiver. The server responds with a <alarmresponse>

The maximum message size for a single telegram is limited to 100 000 characters including XML- header and XML-tags.

8.1 Alarm Requests <alarmrequest>

8.1.1 Element definitions

<reference>

OPTIONAL
1-50 Characters
This id is only a reference number/string. It is not treated by the receiver although it is returned in the <alarmresponse> and is stored in the log for trace purposes.

<authentication>

SUPPLIED BY ALARM OPERATOR
15 Characters
Password for the alarm transmitter

<receiver>

SUPPLIED BY ALARM OPERATOR
1-20 Characters
Distribution information. Determines to which alarm monitoring center the alarm are distributed to if the alarm receiver has several monitoring centers.

<transmittertime>

OPTIONAL

23 Characters.

Ex."2002-05-28 11:35:20.022" Time stamp from the transmitter. Only used for log/trace.

Todo: Contact SOS Alarm and double check if this only supports RFC-822 without timestamp.

<alarmtype>

"OPTIONAL" -> If not present the receiver will assume "AL"

2 Characters

Indicates Alarm or Restore: "AL" = Alarm, "RE" = Restore

<transmittertype>

SUPPLIED BY ALARM OPERATOR

5 Characters

Type of transmitter. Ex: "MC200"

<transmittercode>

SUPPLIED BY ALARM OPERATOR

1-15 Characters

Alarm transmitter number (customer code). Ex: "12345678"

<transmitterarea>

OPTIONAL

1-5 Characters

Different areas on an alarm transmitter can be used to initiate a different action at the alarm receiver on the same alarm code and from the same alarm transmitter

<eventcode>

1-25 Characters

The event code is the carrier of the alarm event. The event codes need to be set up at the alarm operator so that an action will be initiated. The exact event code can technically be anything but it is common to use for example SIA codes, (FA = Fire Alarm, BA = Burglary Alarm)

<section>

OPTIONAL

1-5 Characters

Section identification. Short code for the section where the alarm is active

<sectiontext>

OPTIONAL

1-40 Characters

Section description. Long description of the section where the alarm is active

<detector>

OPTIONAL

1-5 Characters

Detector identification. Short code for the detector that set the alarm active

<detectortext>

OPTIONAL

1-40 Characters

Detector description. Long description of the detector that set the alarm active

<additionalinfo>

OPTIONAL

1-2000 Characters

Additional information about the alarm. Lines are separated by CR+LF or LF; (LF = ASCII 10 (0x0a) and CR= ASCII 13 (0x0d))

<position>

OPTIONAL

n Characters

Contains inner element <pos> that holds the Geographical coordinate.

RT90 (2,5 gon West): “xXXXXXXXyYYYYYYY” where x is the x-coordinate, y is the y- coordinate. Values are given in meters.

Listing 1: RT90

```xml
<position>
   <pos>x1234567y1234567</pos>
</position>
```

WGS84 (Lat/Long): “NDDMMmmEDDDMMmm” where DD are degrees; MM minutes; mm decimal minutes (leading 0 shall be given on the longitude if needed).

Listing 2: WGS84

```xml
<position>
   <pos>N597295E0176288</pos>
</position>
```

Todo: Contact SOS Alarm and clarify what happens when an alarm transmitter has a position in the receiving system but a different one is provided via the alarm.

8.1.2 XML Examples <alarmrequest>

Listing 3: Example of minimum data to send alarm.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<alarmrequest>
  <authentication>hxp4x9nnwxjatv8</authentication>
  <receiver>42</receiver>
  <alarmtype>AL</alarmtype>
  <transmittertype>SV300</transmittertype>
  <transmittercode>1234567</transmittercode>
  <eventcode>BA</eventcode>
</alarmrequest>
```

Listing 4: Example sending alarm using reference.

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<alarmrequest>
   <authentication>hxp4x9nnwxjatv8</authentication>
   <reference>1</reference>
   <receiver>42</receiver>
   <alarmtype>AL</alarmtype>
   <transmittertype>SV300</transmittertype>
   <transmittercode>1234567</transmittercode>
   <eventcode>BA</eventcode>
</alarmrequest>
```

Listing 5: Example of restoring previous sent fire alarm.

```xml
 <?xml version="1.0" encoding="ISO-8859-1"?>
<alarmrequest>
   <authentication>hxp4x9nnwxjatv8</authentication>
   <reference>13843</reference>
   <receiver>42</receiver>
   <transmittertype>SV300</transmittertype>
   <transmittercode>1234567</transmittercode>
   <alarmtype>RE</alarmtype>
   <eventcode>FA</eventcode>
</alarmrequest>^
```

8.2 Alarm Response <alarmresponse>

8.2.1 Element definitions

<reference>

OPTIONAL
1-50 Characters
The transmitter reference from the <alarmrequest> if sent to the receiver.

<status>

Described in *Response Codes*

<info>

Status information in clear text. Also described in *Response Codes*

<arrivaltime>

The time when the receiver received the alarm message.

8.2.2 XML Examples <alarmresponse>

Listing 6: Example of positive response

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<alarmresponse>
    <reference>001</reference>
    <status>0</status>
    <info>OK</info>
    <arrivaltime>2006-12-24 15:00:00.000</arrivaltime>
</alarmresponse>
```

Listing 7: Example of on negative response of a message with wrong authentication

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<alarmresponse>
    <reference>001</reference>
    <status>6</status>
    <info>NOT_AUTHORIZED</info>
    <arrivaltime>2006-12-24 15:00:00</arrivaltime>
</alarmresponse>
```

CHAPTER 9

Requesting new authentication/password

Alarm receiver should implement a function to change the password.

This function should change the password after the first connection to the alarm receiver. The new password shall be stored in the transmitter and used here after. This prevents that and installation company or anybody else knows the password.

This makes it very hard to setup another transmitter to send false alarms.

The new password should not be visible in the configuration interface for the alarm transmitter.

If an error occurs during the password change, the old password will be valid until the first message with the new password is received.

If the NOT_AUTHORIZED reply is received from transmitter the receiver should alert the customer to take proper action. It might be necessary to contact the alarm operators customer support for a new password.

If the transmitter is replaced a new password is required from alarm operator.

The change off password is requested with <requestnewauthentication> and the response <requestnewauthentication-response> is sent back containing the new password.

9.1 New Auth Request <requestnewauthentication>

9.1.1 Element definitions

<authentication>

15 Characters
Authentication (current password)

<reference>

OPTIONAL
1-50 Characters
See *Alarm Requests <alarmrequest>*

<transmittercode>

See *Alarm Requests <alarmrequest>*

<transmittertype>

See *Alarm Requests <alarmrequest>*

9.1.2 XML Examples <requestnewauthentication>

Listing 1: New auth request with reference

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<requestnewauthentication>
    <authentication>14x85dshyrbla27</authentication>
    <reference>46</reference>
    <transmittercode>1234567</transmittercode>
    <transmittertype>ET800</transmittertype>
</requestnewauthentication>
```

9.2 New Auth Response <requestnewauthenticationresponse>

9.2.1 Element definitions

<reference>

1-50 Characters
See *Alarm Requests <alarmrequest>*

<status>

Described in *Response Codes*

<info>

Status information in clear text. Also described in *Response Codes*

<newauthentication>

15 Characters
The new password.

<arrivaltime>

The time when the receiver received the request.

9.2.2 XML Examples <requestnewauthenticationresponse>

Listing 2: New auth response with reference

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<requestnewauthenticationresponse>
    <reference>46</reference>
    <status>0</status>
    <info>OK</info>
```

(continues on next page)

(continued from previous page)

```
    <newauthentication>8usedlb54a234md</newauthentication>
    <arrivaltime>2006-12-24 15:00:00</arrivaltime>
</requestnewauthenticationresponse>
```

CHAPTER 10

Monitored Connection

Monitored connection is a function where the alarm receiver monitors that the transmitter sends heartbeat signals on a regular basis. The alarm transmitter is the initiating part in this function. If the receiver does not get a heartbeat signal in the agreed interval a line fault alarm is generated to the alarm operator.

The service is available in four different levels:

1. 25 hours (90000 seconds)
2. 5 hours (18000 seconds)
3. 180 seconds
4. 90 seconds

When using heartbeat levels 3 and 4 it is mandatory to implement failover functionality in transmitter. If the primary receiver fails to handle the incoming request, the transmitter should resend the alarm to failover receiver.

If the failover receiver fails as well the transmitter shall have alternative delivery way over example GPRS or/and UMTS.

Each service level requires that at least two heartbeat signals shall be sent within the interval.

It is recommend that the transmitter sends at least three heartbeat signals per service level time but no more than six.

Todo: There is a protection if sending <pingrequest> very shortly a after each other that the server responds with PING_TO_OFTEN. What is this exact time limit since it doesn't seem to be bound to the service level? Contact SOS Alarm and ask.

If heartbeat signals is sent to frequent an error message will be replied in the ping request response (PING_TO_OFTEN).

The heartbeat is sent in a <pingrequest> and the server answers with a <pingresponse>

The alarm receiver sends a response message after receiving the ping request. In the response message, the time is attached; this can be used for synchronizing the clock in the transmitter.

10.1 Ping Request <pingrequest>

10.1.1 Element definitions

<authentication>

15 Characters
Authentication (current password)

<reference>

OPTIONAL
1-50 Characters
See *Alarm Requests <alarmrequest>*

<transmittercode>

See *Alarm Requests <alarmrequest>*

<transmittertype>

See *Alarm Requests <alarmrequest>*

10.1.2 XML Examples <pingrequest>

Listing 1: ping request with reference

```
<?xml version="1.0" encoding="iso-8859-1"?>
<pingrequest>
    <authentication>hxp4x9nnwxjatv8</authentication>
    <reference>734632</reference>
    <transmittercode>208013</transmittercode>
    <transmittertype>SV300</transmittertype>
</pingrequest>
```

10.2 Ping Response <pingresponse>

10.2.1 Element definitions

<reference>

1-50 Characters
See *Alarm Requests <alarmrequest>*

<status>

Described in *Response Codes*

<info>

Status information in clear text. Also described in *Response Codes*

<arrivaltime>

The time when the receiver received the alarm message.

10.2.2 XML Examples <requestnewauthenticationresponse>

Listing 2: New ping response with reference

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<pingresponse>
    <reference>734632</reference>
    <status>0</status>
    <info>OK</info>
    <arrivaltime>2005-02-28 11:35:42.012</arrivaltime>
</pingresponse>
```

CHAPTER 11

Response Codes

The following response codes with their status number and info description:

Status	Info	Description
0	OK	OK
1	INVALID_LENGTH	Message to long.
2	INVALID_XML	Invalid xml content.
3	WRONG_CONTENT	Wrong data content, i.e. to long text for a field.
4	NOT_AUTHORIZED	Not authorized, wrong transmitter, instance or password
5	NOT_TREATED_NOT_DISTRIBUTED	Not treated or distributed. Fail over address should be used.
7	MANDATORY_DATA_MISSING	Mandatory XML tag missing
9	SERVICE_UNAVAIVABLE	Not authorized for heartbeat service.
10	DUPLICATED_ALARM	Same alarm received multiple times.
98	SERVER_ERROR	Unknown server error
99	OTHER_ERROR	Unknown receiver error, the transmitter should send alarm to failover address.
100	XML_HEADER_MISSING_OR_INVALID	Invalid or missing XML header.
101	PING_TO_OFTEN	Heartbeat is sent to often.

CHAPTER 12

API Reference

12.1 Client

class sos_access.client.**SOSAccessClient**(*transmitter_code*, *transmitter_type*, *authentication*, *receiver_id*, *receiver_address*, *secondary_receiver_address=None*, *use_single_receiver=False*, *use_tls=False*)

Client implementing the SOS Access v4 protocol to be used for sending Alarm Requests to alarm operators.

Parameters

- **transmitter_code** (*str*) – The code that identifies the transmitter. Supplied by alarm operator
- **transmitter_type** (*str*) – The transmitter type. Supplied by alarm operator
- **authentication** (*str*) – Password for the alarm transmitter. Supplied by alarm operator
- **receiver_id** (*str*) – ID for the receiving system. Needed for the protocol, but might not be needed by the alarm operator.
- **int) receiver_address** (*(str,*) – Tuple of host (IP or FQDN) and port
- **int) secondary_receiver_address** (*(str,*) – Tuple of host (IP or FQDN) and port
- **use_single_receiver** (*bool*) – To enable not supplying secondary_receiver_address
- **use_tls** (*bool*) – Indicates if the transmission of alarm should be encrypted using TLS. This library does not support SSLv3 since it is insecure.

ping(*reference=None*) → sos_access.schemas.PingResponse

Sends a heart beat message to indicate to the alarm operator that the alarm device is still operational

Parameters **reference** (*str*) – Is used as reference for the request and can be searched for in logs.

Returns `PingResponse`

`request_new_auth`(*reference=None*) → sos_access.schemas.NewAuthResponse

Send a request for new password on the server. This is used so that you can have a standard password when deploying devices but it is changed to something the alarm installer does not know when the alarm is operational

Parameters **`reference`** (*`str`*) – Is used as reference for the request and can be searched for in logs.

Returns `NewAuthResponse`

`restore_alarm`(*event_code*, *transmitter_time=None*, *reference=None*, *transmitter_area=None*, *section=None*, *section_text=None*, *detector=None*, *detector_text=None*, *additional_info=None*, *position=None*) → sos_access.schemas.AlarmResponse

Restores an alarm in the receiver.

Parameters

- **`event_code`** (*`str`*) – The event code of the alarm.
- **`transmitter_time`** (*`str`*) – Time of the device or system sending the alarm
- **`reference`** (*`str`*) – A reference that will show up in logs on the alarm receiver.
- **`transmitter_area`** (*`str`*) – Can be used to control different action at the alarm operator on the same transmitter and event_code.
- **`section`** (*`str`*) – Section ID
- **`section_text`** (*`str`*) – Section Description
- **`detector`** (*`str`*) – Detector ID
- **`detector_text`** (*`str`*) – Detector Description
- **`additional_info`** (*`str|dict|list|set|tuple`*) – Extra information about alarm
- **`position`** (*`str`*) – Position data

Returns `AlarmResponse` from alarm receiver

Todo: Is all the extra fields necessary on alarm restore?

`send_alarm`(*event_code*, *transmitter_time=None*, *reference=None*, *transmitter_area=None*, *section=None*, *section_text=None*, *detector=None*, *detector_text=None*, *additional_info=None*, *position=None*) → sos_access.schemas.AlarmResponse

Sends an alarm in the receiver.

Parameters

- **`event_code`** (*`str`*) – The event code of the alarm.
- **`transmitter_time`** (*`datetime.datetime`*) – Time of the device or system sending the alarm
- **`reference`** (*`str`*) – A reference that will show up in logs on the alarm receiver.
- **`transmitter_area`** (*`str`*) – Can be used to control different action at the alarm operator on the same transmitter and event_code.
- **`section`** (*`str`*) – Section ID
- **`section_text`** (*`str`*) – Section Description

- **`detector`** (*`str`*) – Detector ID
- **`detector_text`** (*`str`*) – Detector Description
- **`additional_info`** (*`str|dict|list|set|tuple`*) – Extra information about alarm
- **`position`** (*`str`*) – Position data

Returns `AlarmResponse` from alarm receiver

`transmit`(*data*, *response_schema*, *secondary=False*)
Will create a TCP connection and send the request and received the response and then close the TCP connection.

Parameters

- **`data`** (*`str`*) – The SOS Access XML data to be sent.
- **`response_schema`** (*`AlarmResponseSchema|PingResponseSchema|NewAuthResponseSchem`*
– The schema used to deserialize the response.
- **`secondary`** (*`bool`*) – Indicates if to use the secondary receiver.

`class` `sos_access.client.`**`TCPTransport`**(*address*, *secure=False*, *timeout=5*)
A context manager for TCP sockets to make sure the are closed correctly. Can create both secure and non secures sockets.

`receive`()
Receive data on socket and decode using correct encoding

`send`(*data*)
Send data over socket with correct encoding

12.2 Schemas and models

sos-access uses marshmallow and xmltodict to first make objects into dicts and then into xml when dumping data and the opposit when loading data

`class` `sos_access.schemas.`**`SOSAccessRequest`**
Base SOS Access Request class

`class` `sos_access.schemas.`**`SOSAccessSchema`**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)
Main Schema for serializing and deserializing SOS Access XML data

`class` `sos_access.schemas.`**`AlarmRequest`**(*event_code*, *transmitter_type*, *transmitter_code*, *authentication*, *receiver*, *alarm_type=None*, *transmitter_time=None*, *reference=None*, *transmitter_area=None*, *section=None*, *section_text=None*, *detector=None*, *detector_text=None*, *additional_info=None*, *position=None*)
Represents an AlarmRequest

`additional_info_text`
Additional info is extra info about the alarm. Input is added using the additional_info but we are under

constraint to format the output properly. We accept a list of values. Each item should be printed on a separate line. We use CR+LF to separate lines. If we dont get a list we try to make a string of it and send it on. We must keep the resulting text under 2000 chars. On list input we truncate the last input that makes the message go over the limit On single row we truncate the data and add ... to it.

Returns text

class sos_access.schemas.**AlarmRequestSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a AlarmRequest

class sos_access.schemas.**AlarmResponse**(*status, info, arrival_time=None, reference=None*)

Represents an AlarmResponse

class sos_access.schemas.**AlarmResponseSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a AlarmResponse

class sos_access.schemas.**PingRequest**(*authentication, transmitter_code, transmitter_type, reference=None*)

Represents a PingRequest

class sos_access.schemas.**PingRequestSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a PingRequest

class sos_access.schemas.**PingResponse**(*status, info, arrival_time=None, reference=None*)

Represents a PingResponse

class sos_access.schemas.**PingResponseSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a PingResponse

class sos_access.schemas.**NewAuthRequest**(*authentication, transmitter_code, transmitter_type, reference=None*)

Represents a NewAuthRequest

class `sos_access.schemas.`**`NewAuthRequestSchema`**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a NewAuthRequest

class `sos_access.schemas.`**`NewAuthResponse`**(*status, info, new_authentication=None, arrival_time=None, reference=None*)

Represents a NewAuthResponse

class `sos_access.schemas.`**`NewAuthResponseSchema`**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Dict = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: str = None*)

Schema for dumping and loading a NewAuthResponse

CHAPTER 13

Commercial Support

sos-access is developed by Palmlund Wahlgren Innovative Technology AB. If you need help integration this library we are available for consultation services.

We also run https://alarmer.io , an general integration service that gives you a simple HTTP REST API instead of needing to integrate directly with alarm operators.

CHAPTER 14

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index