secsgem Documentation

Release 0.0.3

Benjamin Parzella

November 23, 2015

Contents

secsgem is a python package to communicate with a host or equipment system in the semiconductor industry.

It is designed to be independent of any external dependencies, making it usable in a closed fab environment without internet connection.

The use cases range from writing tests for implementations or features, simulations in development environments to complete host/equipment implementations. Some parts of the package can be used individually, for example HSMS can be used without SECS-II, or the streams and functions can be used with a different networking stack.

Currently there is no support for communication over serial port (SECS-I, SEMI E04) only ethernet (HSMS, SEMI E37) is available.

HSMS, SECS and GEM are standards from SEMI.

CHAPTER 1

Namespaces

All classes can be accessed with their full module name or directly from the secsgem module.

```
>>> secsgem.format_hex("Hallo")
'48:61:6c:6c:6f'
```

```
>>> secsgem.common.format_hex("Hello")
'48:65:6c:6c:6f'
```

CHAPTER 2

Thanks

- Carl Wolff for his sample TCP socket implementation
- Darius Sullivan for his gist on how to use the streams and functions with twisted

CHAPTER 3

Table of contents

3.1 HSMS

SEMI E37

HSMS defines the communication between host and equipment over the TCP protocol. It specifies packets used to initiate and terminate the connection, check if the link is still active and transfer the actual data.

3.1.1 Packets

A HSMS packet *secsgem.hsms.packets.HsmsPacket* consists of a header *secsgem.hsms.packets.HsmsHeader* and a data part represented by a string. The string contains the additional data encoded as ASCII characters for transmission over TCP. The additional data is only required for a stream/function packet.

```
>>> secsgem.hsms.packets.HsmsPacket(secsgem.hsms.packets.HsmsLinktestReqHeader(2))
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsLinktestReqHeader({'function': 0,
```

Every header has a system id to match the response to a certain request. The system id is the first parameter to the headers constructor. The connection keeps track of the system id, a new one can be requested with the *secsgem.hsms.connections.HsmsConnection.get_next_system_counter()* function.

HSMS packet objects can encode themselves with the *secsgem.hsms.packets.HsmsPacket.encode()* function to a string, which can be sent over the TCP connection.

```
>>> packet = secsgem.hsms.packets.HsmsPacket(secsgem.hsms.packets.HsmsLinktestReqHeader(2))
>>> secsgem.common.format_hex(packet.encode())
'00:00:00:0a:ff:ff:00:00:00:05:00:00:00:02'
```

The other way around, a HSMS packet object can be created from the ASCII string with the *secsgem.hsms.packets.HsmsPacket.decode()* function.

```
>>> secsgem.hsms.packets.HsmsPacket.decode(packetData)
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsHeader({'function': 0, 'stream':
```

There are classes inherited from *secsgem.hsms.packets.HsmsHeader* for all HSMS packet types available:

Type	Class	SType
Select Request	*secsgem.hsms.packets.HsmsSelectReqHeader*	1
Select Response	*secsgem.hsms.packets.HsmsSelectRspHeader*	2
Deselect Request	*secsgem.hsms.packets.HsmsDeselectReqHeader*	3
Deselect Response	*secsgem.hsms.packets.HsmsDeselectRspHeader*	4
Linktest Request	*secsgem.hsms.packets.HsmsLinktestReqHeader*	5
Linktest Response	*secsgem.hsms.packets.HsmsLinktestRspHeader*	6
Reject Request	*secsgem.hsms.packets.HsmsRejectReqHeader*	7
Separate Request	*secsgem.hsms.packets.HsmsSeparateReqHeader*	9
Data Message	*secsgem.hsms.packets.HsmsStreamFunctionHeader*	0

Select Request

Establish HSMS communication

```
>>> secsgem.hsms.packets.HsmsSelectReqHeader(14)
secsgem.hsms.packets.HsmsSelectReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 14, 'sess
```

Select Response

Result of select request

```
>>> secsgem.hsms.packets.HsmsSelectRspHeader(24)
secsgem.hsms.packets.HsmsSelectRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 24, 'sess
```

Deselect Request

Grateful close HSMS communication before disconnecting

```
>>> secsgem.hsms.packets.HsmsDeselectReqHeader(1)
secsgem.hsms.packets.HsmsDeselectReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 1, 'ses
```

Deselect Response

Result of deselect request

```
>>> secsgem.hsms.packets.HsmsDeselectRspHeader(1)
secsgem.hsms.packets.HsmsDeselectRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 1, 'ses
```

Linktest Request

Check the HSMS connection link is good

```
>>> secsgem.hsms.packets.HsmsLinktestReqHeader(2)
secsgem.hsms.packets.HsmsLinktestReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 2, 'ses
```

Linktest Response

Result of linktest request

```
>>> secsgem.hsms.packets.HsmsLinktestRspHeader(10)
secsgem.hsms.packets.HsmsLinktestRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 10, 'se
```

Reject Request

Response to unsupported HSMS message

```
>>> secsgem.hsms.packets.HsmsRejectReqHeader(17, 3, 4)
secsgem.hsms.packets.HsmsRejectReqHeader({'function': 4, 'stream': 3, 'pType': 0, 'system': 17, 'sess
```

Separate Request

Immediate termination of the HSMS connection

```
>>> secsgem.hsms.packets.HsmsSeparateReqHeader(17)
secsgem.hsms.packets.HsmsSeparateReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 17, 'se
```

Data Message

Secs stream and function message

```
>>> secsgem.hsms.packets.HsmsStreamFunctionHeader(22, 1, 1, True, 100)
secsgem.hsms.packets.HsmsStreamFunctionHeader({'function': 1, 'stream': 1, 'pType': 0, 'system': 22,
```

3.1.2 Connections

HSMS has active and passive connections. The active connection is the one making the connection, the passive one is waiting for the incoming connection.

The implementation for the active connection is *secsgem.hsms.connections.HsmsActiveConnection*. For the passive connection there are two implementations:

- *secsgem.hsms.connections.HsmsPassiveConnection* handles only one connection at a time.
- *secsgem.hsms.connections.HsmsMultiPassiveConnection* together with *secsgem.hsms.connections.HsmsMultiPassiveServer* handle multiple connections from different peers.

All connection classes are based on the *secsgem.hsms.connections.HsmsConnection* class, which provides common functionality for all connection types.

The connection process for active and passive connections can be started with the *secsgem.hsms.connections.HsmsPassiveConnection.enable()* function, and stopped with the *secsgem.hsms.connections.HsmsPassiveConnection.disable()* function.

Delegates

All connections work with delegates. When a connection is established/terminated or a packet is received a method of the passed delegate object will be called. The connections support the following delegates:

- on_connection_established(connection)
- on_connection_packet_received(response)

- on_connection_before_closed(connection)
- on_connection_closed(connection)

Sample delegate class:

```
class DelegateSample:
    def on_connection_established(self, connection):
        print "Connection established"

    def on_connection_packet_received(self, connection, packet):
        print "Packet received", packet

    def on_connection_before_closed(self, connection):
        print "Connection about to be terminated"

    def on_connection_closed(self, connection):
        print "Connection terminated"
```

Active connection

For the active connection the first parameter is the IP address of the peer, the second parameter is the port of the peer. The third parameter is the session id the peer is configured for.

Example:

```
>>> delegate = DelegateSample()
>>> conn = secsgem.HsmsActiveConnection('10.211.55.33', 5000, 0, delegate)
>>> conn.enable()
Connection established
Packet received header: {sessionID:0x0000, stream:00, function:04, pType:0x00, sType:0x07, system:0x0
Packet received header: {sessionID:0x0000, stream:00, function:01, pType:0x00, sType:0x07, system:0x0
Connection about to be terminated
Connection terminated
>>> conn.disable()
```

Passive connection

For the passive connection the first parameter is the expected IP address of the peer, the second parameter is the port to listen on. The third parameter is the session id the peer is configured for.

Example:

```
>>> delegate = DelegateSample()
>>> conn = secsgem.HsmsPassiveConnection('10.211.55.33', 5000, 0, delegate)
>>> conn.enable()
Connection established
Packet received header: {sessionID:0xffff, stream:00, function:00, pType:0x00, sType:0x01, system:0x0
Packet received header: {sessionID:0x0000, stream:00, function:03, pType:0x00, sType:0x07, system:0x0
Connection about to be terminated
Connection terminated
>>> conn.disable()
```

Multi-passive connection

In this mode one listening port handles the incoming connections for more than one peer. A instance of *secsgem.hsms.connections.HsmsMultiPassiveServer* is created and connection is created using its

secsgem.hsms.connections.HsmsMultiPassiveServer.create_connection() method. The parameters of the method are the same as for the *Passive connection*. For every available peer a connection must be created using this method.

Example:

```
>>> delegate = DelegateSample()
>>> server = secsgem.HsmsMultiPassiveServer(5000)
>>> conn = server.create_connection('10.211.55.33', 5000, 0, delegate)
>>> conn.enable()
>>> server.start()
Connection established
Packet received header: {sessionID:0xffff, stream:00, function:00, pType:0x00, sType:0x01, system:0x0
Packet received header: {sessionID:0x0000, stream:00, function:03, pType:0x00, sType:0x07, system:0x0
Connection about to be terminated
Connection terminated
>>> conn.disable()
>>> server.stop()
```

Connection manager

The *secsgem.hsms.connectionmanager.HsmsConnectionManager* can be used to manage multiple active and passive connections. It creates and removes *secsgem.hsms.connections.HsmsActiveConnection* and *secsgem.hsms.connections.HsmsMultiPassiveServer*/*secsgem.hsms.connections.HsmsMultiPassiveCo* dynamically.

```
>>> manager=secsgem.HsmsConnectionManager()
>>> handler=manager.add_peer("connection", '10.211.55.33', 5000, False, 0)
>>> handler.enable()
>>> handler.waitfor_linktest_rsp(handler.send_linktest_req())
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsHeader({'function': 0, 'stream':
>>> handler.disable()
>>> manager.stop()
```

Connection manager works with handlers which take care of a lot of the required communication on the matching level (*secsgem.hsms.handler.HsmsHandler*, *secsgem.secs.handler.SecsHandler* and *secsgem.gem.handler.GemHandler*).

3.1.3 Handler

secsgem.hsms.handler.HsmsHandler has the basic HSMS connection handling build in. It automatically selects and deselects the link and performs a periodic linktest. It also replies to incoming HSMS requests like linktest automatically.

```
>>> def on_connect(event, data):
...     print "Connected"
...
>>> client = secsgem.HsmsHandler("10.211.55.33", 5000, False, 0, "test", event_handler=secsgem.EventH
>>> client.enable()
Connected
>>> client.waitfor_linktest_rsp(client.send_linktest_req())
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsHeader({'function': 0, 'stream':
>>> client.disable()
```

The handler has functions to send requests and responses and wait for a certain response.

Events

Events of the handler can be received with the help of *secsgem.common.EventHandler*. The handler sends the following events:

Event name	Description
hsms_connected	Connection was established
hsms_selected	Connection was selected
hsms_disconnected	Connection was terminated

3.2 SECS

SEMI E5

SECS-II defines the messages the data is transferred in between host and equipment over the HSMS protocol (and SECS-I serial). It specifies data types that contain the data and streams and functions that use these types for specific purposes.

3.2.1 Variables

SECS defines a few types to transmit data in.

Data Type	Class	Code
List	*secsgem.secs.variables.SecsVarArray*	L
List	*secsgem.secs.variables.SecsVarList*	L
Binary	*secsgem.secs.variables.SecsVarBinary*	B
Boolean	*secsgem.secs.variables.SecsVarBoolean*	TF
ASCII	*secsgem.secs.variables.SecsVarString*	A
8-Byte integer	*secsgem.secs.variables.SecsVarI8*	I8
1-Byte integer	*secsgem.secs.variables.SecsVarI1*	I1
2-Byte integer	*secsgem.secs.variables.SecsVarI2*	I2
4-Byte integer	*secsgem.secs.variables.SecsVarI4*	I4
8-Byte float	*secsgem.secs.variables.SecsVarF8*	F8
4-Byte float	*secsgem.secs.variables.SecsVarF4*	F8
8-Byte unsigned integer	*secsgem.secs.variables.SecsVarU8*	U8
1-Byte unsigned integer	*secsgem.secs.variables.SecsVarU1*	U1
2-Byte unsigned integer	*secsgem.secs.variables.SecsVarU2*	U2
4-Byte unsigned integer	*secsgem.secs.variables.SecsVarU4*	U4

Example:

```
>>> secsgem.SecsVarString(value="TESTString")
A 'TESTString'
>>> secsgem.SecsVarBoolean(value=True)
TF True
>>> secsgem.SecsVarU4(value=1337)
U4 1337
```

Type arrays

The numeric types can also be an array of that type:

```
>>> secsgem.SecsVarU1(value=[1, 2, 3, 4])
U1 [1, 2, 3, 4]
>>> secsgem.SecsVarBoolean(value=[True, False, False, True])
TF [True, False, False, True]
```

The length of this array can be fixed with the length parameter:

```
>>> secsgem.SecsVarU1(length=3, value=[1, 2, 3])
U1 [1, 2, 3]
>>> secsgem.SecsVarU1(length=3, value=[1, 2, 3, 4])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/bparzella/Developer/secsgem/secsgem/secs/variables.py", line 1694, in __init__
    self.set(value)
  File "/Users/bparzella/Developer/secsgem/secsgem/secs/variables.py", line 1716, in set
    raise ValueError("Value longer than {} chars".format(self.length))
ValueError: Value longer than 3 chars
```

```
>>> secsgem.SecsVarString(length=3, value="Hello").get()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/bparzella/Developer/secsgem/secsgem/secs/variables.py", line 776, in __init__
    self.set(value)
  File "/Users/bparzella/Developer/secsgem/secsgem/secs/variables.py", line 794, in set
    raise ValueError("Value longer than {} chars".format(self.length))
ValueError: Value longer than 3 chars
```

Getting data

The data can be accessed with the *secsgem.secs.variables.SecsVarU1.get()* method, arrays can be accessed using the index operator:

```
>>> secsgem.SecsVarU1(value=1).get()
1
>>> secsgem.SecsVarU1(length=3, value=[1, 2, 3]).get()
[1, 2, 3]
>>> secsgem.SecsVarU1(value=1)[0]
1
>>> secsgem.SecsVarU1(length=3, value=[1, 2, 3])[1]
2
```

Setting data

The data can be set with the *secsgem.secs.variables.SecsVarString.set()* method, arrays can be updated using the index operator:

```
>>> v=secsgem.SecsVarU1(length=3, value=[1, 2, 3])
>>> v.set([3, 2, 1])
>>> v
U1 [3, 2, 1]
>>> v[0] = 1
>>> v
U1 [1, 2, 1]
```

En-/Decoding

The variable types can *secsgem.secs.variables.SecsVarArray.encode()* and *secsgem.secs.variables.SecsVarString.decode()* themselves to ASCII data transferrable with the HSMS protocol:

```
>>> v=secsgem.SecsVarString(value="Hello")
>>> d=v.encode()
>>> d
'A\x05Hello'
>>> secsgem.format_hex(d)
'41:05:48:65:6c:6c:6f'
>>> v.set("NewText")
>>> v
A 'NewText'
>>> v.decode(d)
7
>>> v
A 'Hello'
```

SecsVarArray

secsgem.secs.variables.SecsVarArray is a special type for a list of the same type. The items of the array can be accessed with the index operator.

```
>>> v=secsgem.SecsVarArray(secsgem.SecsVarU4(1))
>>> v.set([1, 2, 3])
>>> v
[U4 1, U4 2, U4 3]
>>> v.get()
[1, 2, 3]
>>> v[1]
2
```

A new item can be appended to the array with the *secsgem.secs.variables.SecsVarArray.append()* method.

SecsVarList

secsgem.secs.variables.SecsVarList is a special type for a list of the different types. The items of the list can be accessed like properties of the object.

An ordered dictionary is required for the creation, because pythons default dictionary will be randomly sorted. Sorting is essential because both peers need to have the data in the same order.

```
>>> v=secsgem.SecsVarList(OrderedDict((
...          ("var1", secsgem.SecsVarU4(1)),
...          ("var2", secsgem.SecsVarString()),
...      )), 2)
>>> v.var1=3
>>> v.var2="Hallo"
>>> v
[var1: U4 3, var2: A 'Hallo']
>>> v.var2
'Hallo'
>>> secsgem.format_hex(v.encode())
'01:02:b1:04:00:00:00:03:41:05:48:61:6c:6c:6f'
```

SecsVarDynamic

secsgem.secs.variables.SecsVarDynamic can take different types, if specified to a certain set of types.

```
>>> v=secsgem.SecsVarDynamic([secsgem.SecsVarString, secsgem.SecsVarU1])
>>> v.set(secsgem.SecsVarString(value="Hello"))
>>> v
A 'Hello'
>>> v.set(secsgem.SecsVarU1(value=10))
>>> v
U1 10
>>> v.set(secsgem.SecsVarU4(value=10))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/bparzella/Developer/secsgem/secsgem/secs/variables.py", line 176, in set
    raise ValueError("Unsupported type {} for this instance of SecsVarDynamic, allowed {}".format(val
ValueError: Unsupported type SecsVarU4 for this instance of SecsVarDynamic, allowed [<class 'secsgem.
```

3.2.2 Functions

A function is inherited from *secsgem.secs.functionbase.SecsStreamFunction*. When inheriting a function only _stream, _function and _formatDescriptor must be overwritten. Evervthing else is implemented in the base class.

Example:

```
class SecsS02F33(secsgem.SecsStreamFunction):
    _stream = 2
    _function = 33

    _formatDescriptor = secsgem.SecsVarList(OrderedDict((
        ("DATAID", secsgem.SecsVarU4(1)),
        ("DATA", secsgem.SecsVarArray(
            secsgem.SecsVarList(OrderedDict((
                ("RPTID", secsgem.SecsVarU4(1)),
                ("VID", secsgem.SecsVarArray(
                    secsgem.SecsVarString()
                )),
            )), 2)
        )),
    )), 2)
```

The data of a function can be read and manipulated with the same functionality as the variables. *secsgem.secs.functionbase.SecsStreamFunction.set()*, *secsgem.secs.functionbase.SecsStreamFunction.get()*, *secsgem.secs.functionbase.SecsStreamFunc* the index operator and object properties. The objects can also en- and decode themselves.

Usage:

```
>>> f=secsgem.SecsS02F33()
>>> f.DATAID=10
>>> f.DATA.append({"RPTID": 5, "VID": ["Hello", "Hallo"]})
>>> f.DATA.append({"RPTID": 6, "VID": ["1", "2"]})
>>> f
S2F33 { [DATAID: U4 10, DATA: [[RPTID: U4 5, VID: [A 'Hello', A 'Hallo']], [RPTID: U4 6, VID: [A '1',
>>> f.DATA[1].VID[0]="Goodbye"
>>> f.DATA[1].VID[1]="Auf Wiedersehen"
>>> f
```

```
S2F33 { [DATAID: U4 10, DATA: [[RPTID: U4 5, VID: [A 'Hello', A 'Hallo']], [RPTID: U4 6, VID: [A 'Goo
>>> secsgem.format_hex(f.encode())
'01:02:b1:04:00:00:00:0a:01:02:01:02:b1:04:00:00:00:05:01:02:41:05:48:65:6c:6c:6f:41:05:48:61:6c:6c:6
```

The encoded data can be used as data string in a *secsgem.hsms.packets.HsmsPacket* together with a *secsgem.hsms.packets.HsmsStreamFunctionHeader*. See Packets.

3.2.3 Handler

secsgem.secs.handler.SecsHandler inherits the functionality from *secsgem.hsms.handler.HsmsHandler* (see Handler).

The SecsHandler has additional functionality to add callbacks for specific streams and functions.

```
>>> def s01f13_handler(connection, packet):
...     print "S1F13 received"
...
>>> def on_connect(event, data):
...     print "Connected"
...
>>> client = secsgem.SecsHandler("10.211.55.33", 5000, False, 0, "test", event_handler=secsgem.EventH
>>> client.register_callback(1, 13, s01f13_handler)
>>>
>>> client.enable()
Connected
S1F13 received
>>> client.disable()
```

There is also additional functionality concerning collection events, service variables and equipment constants.

3.3 GEM

SEMI 30

GEM defines certain behaviors of the equipment and how to use the SECS messages for that purpose.

3.3.1 Handler

secsgem.gem.handler.GemHandler inherits the functionality from *secsgem.secs.handler.SecsHandler* (see Handler).

It automatically handles the whole setup and teardown of the link. Incoming collection events and terminal messages are automatically accepted and propagated by events. The setup of collection event reports is also simplified. It has functionality to send remote commands and handling process programs.

The handler also implements a maintains a communication state, which is defined in the standard.

```
>>> def on_communicating(event, data):
...     print "Communicating"
...
>>> client = secsgem.GemHandler("10.211.55.33", 5000, False, 0, "test", event_handler=secsgem.EventHa
>>>
>>> client.enable()
Communicating
>>> client.get_process_program_list()
```

```
['test1', 'test2']
>>> client.request_process_program('test1')
This is process program test1
>>> client.disable()
```

Also streams/functions can be sent and received with the handler:

```
>>> f = secsgem.SecsS01F01()
>>> client.send_and_waitfor_response(f)
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsHeader({'function': 2, 'stream':
```

Events

GemHandler defines a few new events, that can be received with the help of *secsgem.common.EventHandler*:

Event name	Description
handler_communicating	Connection is setup
collection_event_received	Collection event was received
terminal_received	Terminal message was received

3.4 Class reference

3.4.1 HSMS

Packets

class secsgem.hsms.packets.**HsmsPacket**(*header=None*, *data=''*)

Class for hsms packet.

Contains all required data and functions.

Parameters

- **header** (*secsgem.hsms.packets.HsmsHeader* and derived) – header used for this packet
- **data** (*string*) – data part used for streams and functions (SType 0)

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsPacket(secsgem.hsms.packets.HsmsLinktestReqHeader(2))
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsLinktestReqHeader({'function
```

encode()

Encode packet data to hsms packet

Returns encoded packet

Return type string

Example:

```
>>> import secsgem
>>>
>>> packet = secsgem.hsms.packets.HsmsPacket(secsgem.hsms.packets.HsmsLinktestReqHeader(2))
```

```
>>> secsgem.common.format_hex(packet.encode())
'00:00:00:0a:ff:ff:00:00:00:05:00:00:00:02'
```

static decode (*text*)

Decode byte array hsms packet to HsmsPacket object

Returns received packet object

Return type *secsgem.hsms.packets.HsmsPacket*

Example:

```
>>> import secsgem
>>>
>>> packetData = "\x00\x00\x00\x0b\xff\xff\x00\x00\x00\x05\x00\x00\x00\x02"
>>>
>>> secsgem.format_hex(packetData)
'00:00:00:0b:ff:ff:00:00:00:05:00:00:00:02'
>>>
>>> secsgem.hsms.packets.HsmsPacket.decode(packetData)
secsgem.hsms.packets.HsmsPacket({'header': secsgem.hsms.packets.HsmsHeader({'function': 0, '
```

class secsgem.hsms.packets.**HsmsHeader** (*system*, *session_id*)

Generic HSMS header

Base for different specific headers

Parameters

- **system** (*integer*) – message ID
- **session_id** (*integer*) – device / session ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsHeader(3, 100)
secsgem.hsms.packets.HsmsHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 3, 'sessionID
```

class secsgem.hsms.packets.**HsmsStreamFunctionHeader** (*system*, *stream*, *function*, *require_response*, *session_id*)

Bases: *secsgem.hsms.packets.HsmsHeader*

Header for SECS message

Header for message with SType 0.

Parameters

- **system** (*integer*) – message ID
- **stream** (*integer*) – messages stream
- **function** (*integer*) – messages function
- **require_response** (*boolean*) – is response expected from remote
- **session_id** (*integer*) – device / session ID

Example:

```
>>> import secsgem
>>>
```

```
>>> secsgem.hsms.packets.HsmsStreamFunctionHeader(22, 1, 1, True, 100)
secsgem.hsms.packets.HsmsStreamFunctionHeader({'function': 1, 'stream': 1, 'pType': 0, 'system':
```

class secsgem.hsms.packets.**HsmsSelectReqHeader**(*system*)

Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Select Request

Header for message with SType 1.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsSelectReqHeader(14)
secsgem.hsms.packets.HsmsSelectReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 14,
```

class secsgem.hsms.packets.**HsmsSelectRspHeader**(*system*)

Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Select Response

Header for message with SType 2.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsSelectRspHeader(24)
secsgem.hsms.packets.HsmsSelectRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 24,
```

class secsgem.hsms.packets.**HsmsDeselectReqHeader**(*system*)

Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Deselect Request

Header for message with SType 3.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsDeselectReqHeader(1)
secsgem.hsms.packets.HsmsDeselectReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 1,
```

class secsgem.hsms.packets.**HsmsDeselectRspHeader**(*system*)

Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Deselect Response

Header for message with SType 4.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsDeselectRspHeader(1)
secsgem.hsms.packets.HsmsDeselectRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 1,
```

class secsgem.hsms.packets.**HsmsLinktestReqHeader**(*system*)
Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Linktest Request

Header for message with SType 5.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsLinktestReqHeader(2)
secsgem.hsms.packets.HsmsLinktestReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 2,
```

class secsgem.hsms.packets.**HsmsLinktestRspHeader**(*system*)
Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Linktest Response

Header for message with SType 6.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsLinktestRspHeader(10)
secsgem.hsms.packets.HsmsLinktestRspHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 10
```

class secsgem.hsms.packets.**HsmsRejectReqHeader**(*system*, *s_type*, *reason*)
Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Reject Request

Header for message with SType 7.

Parameters

- **system** (*integer*) – message ID
- **s_type** (*integer*) – sType of rejected message
- **reason** (*integer*) – reason for rejection

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsRejectReqHeader(17, 3, 4)
secsgem.hsms.packets.HsmsRejectReqHeader({'function': 4, 'stream': 3, 'pType': 0, 'system': 17,
```

class secsgem.hsms.packets.**HsmsSeparateReqHeader**(*system*)
Bases: *secsgem.hsms.packets.HsmsHeader*

Header for Separate Request

Header for message with SType 9.

Parameters **system** (*integer*) – message ID

Example:

```
>>> import secsgem
>>>
>>> secsgem.hsms.packets.HsmsSeparateReqHeader(17)
secsgem.hsms.packets.HsmsSeparateReqHeader({'function': 0, 'stream': 0, 'pType': 0, 'system': 17
```

Connections

class secsgem.hsms.connections.**HsmsConnection**(*active*, *address*, *port*, *session_id=0*, *delegate=None*)

Bases: object

Connection class used for active and passive hsms connections.

Parameters

- **active** (*boolean*) – Is the connection active (*True*) or passive (*False*)
- **address** (*string*) – IP address of remote host
- **port** (*integer*) – TCP port of remote host
- **session_id** (*integer*) – session / device ID to use for connection
- **delegate** (inherited from *secsgem.hsms.handler.HsmsHandler*) – target for messages

selectTimeout = 0.5
Timeout for select calls

sendBlockSize = 1048576
Block size for outbound data

T3 = 45.0
Reply Timeout

T5 = 10.0
Connect Separation Time

T6 = 5.0
Control Transaction Timeout

disconnect()
Close connection

send_packet(*packet*)
Send the ASCII coded packet to the remote host

Parameters **packet** (*string / byte array*) – encoded data to be transmitted

get_next_system_counter()
Returns the next System.

Returns System for the next command

Return type integer

class secsgem.hsms.connections.**HsmsActiveConnection**(*address*, *port=5000*, *session_id=0*, *delegate=None*)

Bases: *secsgem.hsms.connections.HsmsConnection*

Client class for single active (outgoing) connection

Parameters

- **address** (*string*) – IP address of target host
- **port** (*integer*) – TCP port of target host
- **session_id** (*integer*) – session / device ID to use for connection
- **delegate** (*object*) – target for messages

Example:

```
# TODO: create example
```

T3 = 45.0

T5 = 10.0

T6 = 5.0

disconnect()
Close connection

enable()
Enable the connection.

Starts the connection process to the passive remote.

get_next_system_counter()
Returns the next System.

Returns System for the next command

Return type integer

selectTimeout = 0.5

sendBlockSize = 1048576

send_packet(*packet*)
Send the ASCII coded packet to the remote host

Parameters **packet** (*string / byte array*) – encoded data to be transmitted

disable()
Disable the connection.

Stops all connection attempts, and closes the connection

class secsgem.hsms.connections.**HsmsPassiveConnection**(*address*, *port=5000*, *session_id=0*, *delegate=None*)
Bases: *secsgem.hsms.connections.HsmsConnection*

Server class for single passive (incoming) connection

Creates a listening socket and waits for one incoming connection on this socket. After the connection is established the listening socket is closed.

Parameters

- **address** (*string*) – IP address of target host
- **port** (*integer*) – TCP port of target host
- **session_id** (*integer*) – session / device ID to use for connection
- **delegate** (*object*) – target for messages

Example:

```
# TODO: create example
```

enable()
Enable the connection.

Starts the connection process to the passive remote.

disable()
Disable the connection.

Stops all connection attempts, and closes the connection

T3 = 45.0

T5 = 10.0

T6 = 5.0

disconnect()
Close connection

get_next_system_counter()
Returns the next System.

Returns System for the next command

Return type integer

selectTimeout = 0.5

sendBlockSize = 1048576

send_packet(*packet*)
Send the ASCII coded packet to the remote host

Parameters **packet** (*string / byte array*) – encoded data to be transmitted

class secsgem.hsms.connections.**HsmsMultiPassiveConnection**(*address*, *port=5000*, *session_id=0*, *delegate=None*)
Bases: *secsgem.hsms.connections.HsmsConnection*

Connection class for single connection from *secsgem.hsms.connections.HsmsMultiPassiveServer*

Handles connections incoming connection from *secsgem.hsms.connections.HsmsMultiPassiveServer*

Parameters

- **address** (*string*) – IP address of target host
- **port** (*integer*) – TCP port of target host
- **session_id** (*integer*) – session / device ID to use for connection
- **delegate** (*object*) – target for messages

Example:

```
# TODO: create example
```

on_connected(*sock*, *address*)
Connected callback for *secsgem.hsms.connections.HsmsMultiPassiveServer*

Parameters

- **sock** (Socket) – Socket for new connection

- **address** (*string*) – IP address of remote host

enable()
Enable the connection.

Starts the connection process to the passive remote.

disable()
Disable the connection.

Stops all connection attempts, and closes the connection

T3 = 45.0

T5 = 10.0

T6 = 5.0

disconnect()
Close connection

get_next_system_counter()
Returns the next System.

Returns System for the next command

Return type integer

selectTimeout = 0.5

sendBlockSize = 1048576

send_packet(*packet*)
Send the ASCII coded packet to the remote host

Parameters **packet** (*string / byte array*) – encoded data to be transmitted

class secsgem.hsms.connections.**HsmsMultiPassiveServer**(*port=5000*)
Bases: object

Server class for multiple passive (incoming) connection. The server creates a listening socket and waits for incoming connections on this socket.

Parameters **port** (*integer*) – TCP port to listen on

Example:

```
# TODO: create example
```

selectTimeout = 0.5
Timeout for select calls

create_connection(*address*, *port=5000*, *session_id=0*, *delegate=None*)
Create and remember connection for the server

Parameters

- **address** (*string*) – IP address of target host
- **port** (*integer*) – TCP port of target host
- **session_id** (*integer*) – session / device ID to use for connection
- **delegate** (*object*) – target for messages

start()
Starts the server and returns. It will launch a listener running in background to wait for incoming connections.

stop(*terminate_connections=True*)
Stops the server. The background job waiting for incoming connections will be terminated. Optionally all connections received will be closed.

Parameters **terminate_connections** (*boolean*) – terminate all connection made by this server

Handler

class secsgem.hsms.handler.**HsmsHandler**(*address*, *port*, *active*, *session_id*, *name*, *event_handler=None*, *custom_connection_handler=None*)
Bases: *secsgem.common.EventProducer*

Baseclass for creating Host/Equipment models. This layer contains the HSMS functionality. Inherit from this class and override required functions.

Parameters

- **address** (*string*) – IP address of remote host
- **port** (*integer*) – TCP port of remote host
- **active** (*boolean*) – Is the connection active (*True*) or passive (*False*)
- **session_id** (*integer*) – session / device ID to use for connection
- **name** (*string*) – Name of the underlying configuration
- **event_handler** (*secsgem.common.EventHandler*) – object for event handling
- **custom_connection_handler** (*secsgem.hsms.connections.HsmsMultiPassiveServer*) – object for connection handling (ie multi server)

Example:

```
import secsgem

def onConnect(event, data):
    print "Connected"

client = secsgem.HsmsHandler("10.211.55.33", 5000, True, 0, "test", event_handler=secsgem.EventH

client.enable()

time.sleep(3)

client.disable()
```

on_connection_established(*_*)
Connection was established

on_connection_before_closed(*_*)
Connection is about to be closed

on_connection_closed(*_*)
Connection was closed

on_connection_packet_received(*_*, *packet*)
Packet received by connection

Parameters **packet** (*secsgem.hsms.packets.HsmsPacket*) – received data packet

enable()
Enables the connection

disable()
Disables the connection

waitfor_stream_function(*stream*, *function*, *is_control=False*)
Wait for an incoming stream and function and return the receive data

Parameters

- **stream** (*integer*) – number of stream to wait for
- **function** (*integer*) – number of function to wait for
- **is_control** (*bool*) – is it a control packet

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_stream_function(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

waitfor_system(*system*, *is_control=False*)
Wait for an message with supplied system

Parameters **system** (*integer*) – number of system to wait for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_and_waitfor_response(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_response(*function*, *system*)
Send response function for system

Parameters

- **function** (*secsgem.secs.functionbase.SecsStreamFunction*) – function to be sent
- **system** (*integer*) – system to reply to

send_select_req()
Send a Select Request to the remote host

Returns System of the sent request

Return type integer

send_select_rsp(*system_id*)
Send a Select Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

waitfor_select_rsp(*system_id*)
Wait for an incoming Select Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_linktest_req()
Send a Linktest Request to the remote host

Returns System of the sent request

Return type integer

send_linktest_rsp(*system_id*)
Send a Linktest Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

waitfor_linktest_rsp(*system_id*)
Wait for an incoming Linktest Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_deselect_req()
Send a Deselect Request to the remote host

Returns System of the sent request

Return type integer

send_deselect_rsp(*system_id*)
Send a Deselect Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

fire_event(*event_name*, *data*, *async=False*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **data** (*dict*) – parameters for event

waitfor_deselect_rsp(*system_id*)
Wait for an incoming Deselect Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_reject_rsp(*system_id*, *s_type*, *reason*)
Send a Reject Response to the remote host

Parameters

- **system_id** (*integer*) – System of the request to reply for
- **s_type** (*integer*) – s_type of rejected message

- **reason** (*integer*) – reason for rejection

send_separate_req()
Send a Separate Request to the remote host

ConnectionManager

class secsgem.hsms.connectionmanager.**HsmsConnectionManager**(*event_handler=None*)
Bases: *secsgem.common.EventProducer*

High level class that handles multiple active and passive connections and the model for them.

Parameters **event_handler** (*secsgem.common.EventHandler*) – object for event handling

has_connection_to(*index*)
Check if connection to certain peer exists.

Parameters **index** (*string*) – Name of the reqested handler.

Returns Is peer available

Return type boolean

static get_connection_id(*address*)
Generates connection ids used for internal indexing.

Parameters **address** (*string*) – The IP address for the affected remote.

add_peer(*name*, *address*, *port*, *active*, *session_id*, *connection_handler=<class secsgem.hsms.handler.HsmsHandler>*)
Add a new connection

Parameters

- **name** (*string*) – Name of the peers configuration
- **address** (*string*) – IP address of peer
- **port** (*integer*) – TCP port of peer
- **active** (*boolean*) – Is the connection active (*True*) or passive (*False*)
- **session_id** (*integer*) – session / device ID of peer
- **connection_handler** (inherited from *secsgem.hsms.handler.HsmsHandler*) – Model handling this connection

remove_peer(*name*, *address*, *port*)
Remove a previously added connection

Parameters

- **name** (*string*) – Name of the peers configuration
- **address** (*string*) – IP address of peer
- **port** (*integer*) – TCP port of peer

stop()
Stop all servers and terminate the connections

fire_event(*event_name*, *data*, *async=False*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **data** (*dict*) – parameters for event

3.4.2 SECS

Variables

SECS variable types

class secsgem.secs.variables.**SecsVar**
Bases: object

Base class for SECS variables. Due to the python types, wrapper classes for variables are required. If constructor is called with SecsVar or subclass only the value is copied.

formatCode = -1

set(*value*)
Set the internal value to the provided value

Parameters **value** (*various*) – new value

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

class secsgem.secs.variables.**SecsVarDynamic**(*types*, *length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Variable with interchangable type.

Parameters

- **types** (list of *secsgem.secs.variables.SecsVar* classes) – list of supported types, default first. empty list means all types are support, SecsVarString default
- **length** (*integer*) – max number of items in type
- **value** (*various*) – initial value

set(*value*)
Set the internal value to the provided value

In doubt provide the variable wrapped in the matching *secsgem.secs.variables.SecsVar* class, to avoid confusion.

Example:

```
>>> import secsgem
>>>
>>> var = secsgem.SecsVarDynamic([secsgem.SecsVarString, secsgem.SecsVarU1])
>>> var.set(10)
>>> var
A '10'
>>> var.set(secsgem.SecsVarU1(value=10))
>>> var
U1 10
```

If no type is provided the default type is used which might not be the expected type.

Parameters **value** (*various*) – new value

get()
Return the internal value

Returns internal value

Return type various

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarDynamic*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

formatCode = -1

class secsgem.secs.variables.**SecsVarList**(*data*, *field_count=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

List variable type. List with items of different types

Parameters

- **data** (*OrderedDict*) – internal data values
- **field_count** (*integer*) – number of fields in the list
- **value** (*dict/list*) – initial value

formatCode = 0

set(*value*)
Set the internal value to the provided value

Parameters **value** (*dict/list*) – new value

get()
Return the internal value

Returns internal value

Return type list

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarList*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarArray**(*data*, *field_count=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

List variable type. List with items of same type

Parameters

- **data** (*secsgem.secs.variables.SecsVar*) – internal data definition/sample
- **field_count** (*integer*) – number of fields in the list
- **value** (*list*) – initial value

formatCode = 0

append(*data*)
Append data to the internal list

Parameters **value** (*various*) – new value

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list*) – new value

get()
Return the internal value

Returns internal value

Return type list

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarArray*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarBinary**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for binary data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*string/integer*) – initial value

formatCode = 8

set(*value*)
Set the internal value to the provided value

Parameters **value** (*string/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarBinary*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarBoolean**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for boolean data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/boolean*) – initial value

formatCode = 9

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/boolean*) – new value

get()
Return the internal value

Returns internal value

Return type list/boolean

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarBoolean*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarString**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for string data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*string*) – initial value

formatCode = 16

set(*value*)
Set the internal value to the provided value

Parameters **value** (*string*) – new value

get()
Return the internal value

Returns internal value

Return type string

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarString*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarI8**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 8 byte signed data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 24

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode (*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone ()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarI8*

decode_item_header (*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header (*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarI1** (*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 1 byte signed data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 25

set (*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get ()
Return the internal value

Returns internal value

Return type list/integer

encode ()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarI1*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarI2**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 2 byte signed data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 26

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarI2*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarI4**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 4 byte signed data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 28

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarI4*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarF8**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 8 byte float data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/float*) – initial value

formatCode = 32

set (*value*)
Set the internal value to the provided value

Parameters **value** (*list/float*) – new value

get ()
Return the internal value

Returns internal value

Return type list/float

encode ()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode (*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone ()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarF8*

decode_item_header (*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header (*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarF4** (*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 4 byte float data

Parameters

- **length** (*integer*) – number of items this value

- **value** (*list/float*) – initial value

formatCode = 36

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/float*) – new value

get()
Return the internal value

Returns internal value

Return type list/float

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarF4*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarU8**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 8 byte unsigned data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 40

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarU8*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarU1**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 1 byte unsigned data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 41

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarU1*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarU2**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 2 byte unsigned data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 42

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarU2*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

class secsgem.secs.variables.**SecsVarU4**(*length=-1*, *value=None*)
Bases: *secsgem.secs.variables.SecsVar*

Secs type for 4 byte unsigned data

Parameters

- **length** (*integer*) – number of items this value
- **value** (*list/integer*) – initial value

formatCode = 44

set(*value*)
Set the internal value to the provided value

Parameters **value** (*list/integer*) – new value

get()
Return the internal value

Returns internal value

Return type list/integer

encode()
Encode the value to secs data

Returns encoded data bytes

Return type string

decode(*data*, *start=0*)
Decode the secs byte data to the value

Parameters

- **data** (*string*) – encoded data bytes
- **start** (*integer*) – start position of value the data

Returns new start position

Return type integer

clone()
Returns copy of the object

Returns copy

Return type *secsgem.secs.variables.SecsVarU4*

decode_item_header(*data*, *text_pos=0*)
Encode item header depending on the number of length bytes required.

Parameters

- **data** (*string*) – encoded data
- **text_pos** (*integer*) – start of item header in data

Returns start position for next item, format code, length item of data

Return type (integer, integer, integer)

encode_item_header(*length*)
Encode item header depending on the number of length bytes required.

Parameters **length** (*integer*) – number of bytes in data

Returns encoded item header bytes

Return type string

FunctionBase

Base class for for SECS stream and functions

class secsgem.secs.functionbase.**SecsStreamFunction**(*value=None*)
Bases: object

Secs stream and function base class

This class is inherited to create a stream/function class. To create a function specific content the class variables _stream, _function and _formatDescriptor must be overridden.

Example:

```
class SecsS02F30(SecsStreamFunction):
    _stream = 2
    _function = 30

    _formatDescriptor = SecsVarArray(SecsVarList(OrderedDict((
                        ("ECID", SecsVarU4(1)),
                        ("ECNAME", SecsVarString()),
                        ("ECMIN", SecsVarDynamic([SecsVarString])),
                        ("ECMAX", SecsVarDynamic([SecsVarString])),
                        ("ECDEF", SecsVarDynamic([SecsVarString])),
                        ("UNITS", SecsVarString()),
                       )), 6))
```

Parameters **value** (*various*) – set the value of stream/function parameters

append(*data*)
Append data to list, if stream/function parameter is a list

Parameters **data** (*various*) – list item to add

encode()
Generates the encoded hsms data of the stream/function parameter

Returns encoded data

Return type string

decode(*data*)
Updates stream/function parameter data from the passed data

Parameters **data** (*string*) – encoded data

set(*value*)
Updates the value of the stream/function parameter

Parameters **value** (*various*) – new value for the parameter

get()
Gets the current value of the stream/function parameter

Returns current parameter value

Return type various

Functions

Wrappers for SECS stream and functions

class secsgem.secs.functions.**SecsS00F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 00, function 00 - hsms communication

Example:

```
>>> import secsgem
>>> secsgem.SecsS00F00()
S0F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS01F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 00 - abort transaction stream 1

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F00()
S1F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS01F01**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 01 - are you online - request

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F01()
S1F1 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS01F02E**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 02 - on line data (Equipment)

Structure:

```
{
    MDLN: A[20]
    SOFTREV: A[20]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F02E({"MDLN": "secsgem", "SOFTREV": "0.0.3"})
S1F2 { [MDLN: A 'secsgem', SOFTREV: A '0.0.3'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F02H**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 02 - on line data (Host)

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F02H()
S1F2 { [] }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS01F03**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 03 - selected equipment status - request

Structure:

```
[
    SVID: U4[1]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F03([1, 6, 12])
S1F3 { [U4 1, U4 6, U4 12] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F04**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 04 - selected equipment status - data

Structure:

```
[
    SV: various
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F04([secsgem.SecsVarU1(value=1), "text", secsgem.SecsVarU4(value=1337)])
S1F4 { [U1 1, A 'text', U4 1337] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F11**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 11 - status variable namelist - request

Structure:

```
[
    SVID: U4[1]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F11([1, 1337])
S1F11 { [U4 1, U4 1337] }
```

An empty list will return all available status variables.

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F12**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 12 - status variable namelist - reply

Structure:

```
[
    {
        SVID: U4[1]
        SVNAME: A[n]
        UNITS: A[n]
    }
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F12([{"SVID": 1, "SVNAME": "SV1", "UNITS": "mm"}, {"SVID": 1337, "SVNAME": "S
S1F12 { [[SVID: U4 1, SVNAME: A 'SV1', UNITS: A 'mm'], [SVID: U4 1337, SVNAME: A 'SV2', UNITS: A
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F13E**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 13 - establish communication - request (Equipment)

Structure:

```
{
    MDLN: A[20]
    SOFTREV: A[20]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F13E({"MDLN": "secsgem", "SOFTREV": "0.0.3"})
S1F13 { [MDLN: A 'secsgem', SOFTREV: A '0.0.3'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F13H**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 13 - establish communication - request (Host)

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F13H()
S1F13 { [] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F14E**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 14 - establish communication - acknowledge (Equipment)

Structure:

```
{
    COMMACK: B[1]
    DATA: {
        MDLN: A[20]
        SOFTREV: A[20]
    }
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F14E({"COMMACK": 1, "DATA": {"MDLN": "secsgem", "SOFTREV": "0.0.3"}})
S1F14 { [COMMACK: B 1, DATA: [MDLN: A 'secsgem', SOFTREV: A '0.0.3']] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS01F14H**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 01, function 14 - establish communication - acknowledge (Host)

Structure:

```
{
    COMMACK: B[1]
    DATA: {
```

```
        }
    }
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS01F14H({"COMMACK": 1, "DATA": {}})
S1F14 { [COMMACK: B 1, DATA: []] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 00 - abort transaction stream 2

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F00()
S2F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS02F13**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 13 - equipment constant - request

Structure:

```
[
    ECID: U4[1]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F13([1, 1337])
S2F13 { [U4 1, U4 1337] }
```

An empty list will return all available equipment constants.

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F14**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 14 - equipment constant - data

Structure:

```
[
    ECV: various
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F14([secsgem.SecsVarU1(value=1), "text"])
S2F14 { [U1 1, A 'text'] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F15**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 15 - new equipment constant - send

Structure:

```
[
    {
        ECID: U4[1]
        ECV: various
    }
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F15([{"ECID": 1, "ECV": secsgem.SecsVarU4(value=10)}, {"ECID": 1337, "ECV": "
S2F15 { [[ECID: U4 1, ECV: U4 10], [ECID: U4 1337, ECV: A 'text']] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F16**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 16 - new equipment constant - acknowledge

Structure:

```
EAC: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F16(1)
S2F16 { B 1 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F29**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 29 - equipment constant namelist - request

Structure:

```
[
    ECID: U4[1]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F29([1, 1337])
S2F29 { [U4 1, U4 1337] }
```

An empty list will return all available equipment constants.

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F30**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 30 - equipment constant namelist

Structure:

```
[
    {
        ECID: U4[1]
        ECNAME: A[n]
        ECMIN: various
        ECMAX: various
        ECDEF: various
        UNITS: A[n]
    }
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F30([{"ECID": 1, "ECNAME": "EC1", "ECMIN": secsgem.SecsVarU1(value=0), "ECMAX
S2F30 { [[ECID: U4 1, ECNAME: A 'EC1', ECMIN: U1 0, ECMAX: U1 100, ECDEF: U1 50, UNITS: A 'mm'],
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F33**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 33 - define report

Structure:

```
{
    DATAID: U4[1]
    DATA: [
        {
            RPTID: U4[1]
            RPT: [
                VID: A[n]
                ...
            ]
        }
        ...
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F33({"DATAID": 1, "DATA": [{"RPTID": 1000, "VID": [12, 1337]}, {"RPTID": 1001
S2F33 { [DATAID: U4 1, DATA: [[RPTID: U4 1000, VID: [A '12', A '1337']], [RPTID: U4 1001, VID: [
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F34**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 34 - define report - acknowledge

Structure:

```
DRACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F34(0)
S2F34 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F35**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 35 - link event report

Structure:

```
{
    DATAID: U4[1]
    DATA: [
        {
            CEID: U4[1]
            RPTID: [
                ID: U4[1]
                ...
            ]
        }
        ...
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F35({"DATAID": 1, "DATA": [{"CEID": 1337, "RPTID": [1000, 1001]}]})
S2F35 { [DATAID: U4 1, DATA: [[CEID: U4 1337, RPTID: [U4 1000, U4 1001]]]] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F36**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 36 - link event report - acknowledge

Structure:

```
LRACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F36(0)
S2F36 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F37**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 37 - en-/disable event report

Structure:

```
{
    CEED: BOOL[1]
    CEID: [
        ID: U4[1]
        ...
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F37({"CEED": True, "CEID": [1337]})
S2F37 { [CEED: TF True, CEID: [U4 1337]] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F38**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 38 - en-/disable event report - acknowledge

Structure:

```
ERACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F38(0)
S2F38 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F41**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 41 - host command - send

Structure:

```
{
    RCMD: A[n]
    PARAMS: [
        {
            CPNAME: A[n]
            CPVAL: A[n]
```

```
            }
            ...
        ]
    }
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F41({"RCMD": "COMMAND", "PARAMS": [{"CPNAME": "PARAM1", "CPVAL": "VAL1"}, {"C
S2F41 { [RCMD: A 'COMMAND', PARAMS: [[CPNAME: A 'PARAM1', CPVAL: A 'VAL1'], [CPNAME: A 'PARAM2',
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS02F42**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 02, function 42 - host command - acknowledge

Structure:

```
{
    HCACK: B[1]
    PARAMS: [
        {
            CPNAME: A[n]
            CPACK: B[1]
        }
        ...
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS02F42({"HCACK": 1, "PARAMS": [{"CPNAME": "PARAM1", "CPACK": 2}, {"CPNAME": "PAR
S2F42 { [HCACK: B 1, PARAMS: [[CPNAME: A 'PARAM1', CPACK: B 2], [CPNAME: A 'PARAM2', CPACK: B 3]
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS05F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 05, function 00 - abort transaction stream 5

Example:

```
>>> import secsgem
>>> secsgem.SecsS05F00()
S5F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS05F01**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 05, function 01 - alarm report - send

Structure:

```
{
    ALCD: B[1]
    ALID: U4[1]
    ALTX: A[120]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS05F01({"ALCD": 1, "ALID": 100, "ALTX": "text"})
S5F1 { [ALCD: B 1, ALID: U4 100, ALTX: A 'text'] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS05F02**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 05, function 02 - alarm report - acknowledge

Structure:

```
ACKC5: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS05F02(0)
S5F02 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS06F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 06, function 00 - abort transaction stream 6

Example:

```
>>> import secsgem
>>> secsgem.SecsS06F00()
S6F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS06F11**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 06, function 11 - event report

Structure:

```
{
    DATAID: U4[1]
    CEID: U4[1]
    RPT: [
        {
            RPTID: U4[1]
            V: [
                DATA: various
```

```
                    ...
                ]
            }
            ...
        ]
    }
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS06F11({"DATAID": 1, "CEID": 1337, "RPT": [{"RPTID": 1000, "V": ["VAR", secsgem.
S6F11 { [DATAID: U4 1, CEID: U4 1337, RPT: [[RPTID: U4 1000, V: [A 'VAR', U4 100]]]] }
```

Parameters **value** (*list*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS06F12**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 06, function 12 - event report - acknowledge

Structure:

```
ACKC6: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS06F12(0)
S6F12 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F00**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 00 - abort transaction stream 7

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F00()
S7F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS07F01**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 01 - process program load - inquire

Structure:

```
{
    PPID: A[n]
    LENGTH: U4[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F01({"PPID": "program", "LENGTH": 4})
S7F1 { [PPID: A 'program', LENGTH: U4 4] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F02**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 02 - process program load - grant

Structure:

```
PPGNT: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F02(0)
S7F2 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F03**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 03 - process program - send

Structure:

```
{
    PPID: A[n]
    PPBODY: B[n]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F03({"PPID": "program", "PPBODY": "data"})
S7F3 { [PPID: A 'program', PPBODY: B <4 bytes>] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F04**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 04 - process program - acknowledge

Structure:

```
ACKC7: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F04(0)
S7F4 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F05**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 05 - process program - request

Structure:

```
PPID: A[n]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F05("program")
S7F5 { A 'program' }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F06**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 06 - process program - data

Structure:

```
{
    PPID: A[n]
    PPBODY: B[n]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F06({"PPID": "program", "PPBODY": "data"})
S7F6 { [PPID: A 'program', PPBODY: B <4 bytes>] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F17**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 17 - delete process program - send

Structure:

```
[
    PPID: A[n]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F17(["program1", "program2"])
S7F17 { [A 'program1', A 'program2'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F18**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 18 - delete process program - acknowledge

Structure:

```
ACKC7: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F18(0)
S7F18 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F19**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 19 - current equipment process program - request

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F19()
S7F19 { None }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS07F20**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 07, function 20 - current equipment process program - data

Structure:

```
[
    PPID: A[n]
    ...
]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS07F20(["program1", "program2"])
S7F20 { [A 'program1', A 'program2'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F00**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 00 - abort transaction stream 9

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F00()
S9F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS09F01**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 01 - unrecognized device id

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F01("HEADERDATA")
S9F1 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F03**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 03 - unrecognized stream type

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F03("HEADERDATA")
S9F3 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F05**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 05 - unrecognized function type

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F05("HEADERDATA")
S9F5 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F07**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 07 - illegal data

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F07("HEADERDATA")
S9F7 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F09**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 09 - transaction timer timeout

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F09("HEADERDATA")
S9F9 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F11**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 11 - data too long

Structure:

```
MHEAD: B[10]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F11("HEADERDATA")
S9F11 { B <10 bytes> }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS09F13**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 09, function 13 - conversation timeout

Structure:

```
{
    MEXP: A[6]
    EDID: A[80]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS09F13({"MEXP": "S01E01", "EDID": "data"})
S9F13 { [MEXP: A 'S01E01', EDID: A 'data'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS10F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 10, function 00 - abort transaction stream 10

Example:

```
>>> import secsgem
>>> secsgem.SecsS10F00()
S10F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS10F01**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 10, function 01 - terminal - request

Structure:

```
{
    TID: B[1]
    TEXT: A[]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS10F01({"TID": 0, "TEXT": "hello?"})
S10F1 { [TID: B 0, TEXT: A 'hello?'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS10F02**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 10, function 02 - terminal - acknowledge

Structure:

```
ACK10: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS10F02(0)
S10F2 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS10F03**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 10, function 03 - terminal single - display

Structure:

```
{
    TID: B[1]
    TEXT: A[]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS10F03({"TID": 0, "TEXT": "hello!"})
S10F3 { [TID: B 0, TEXT: A 'hello!'] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS10F04**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 10, function 04 - terminal single - acknowledge

Structure:

```
ACK10: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS10F04(0)
S10F4 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F00**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 00 - abort transaction stream 12

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F00()
S12F0 { None }
```

Parameters **value** (*None*) – function has no parameters

class secsgem.secs.functions.**SecsS12F01**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 01 - map setup data - send

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    FNLOC: U2[1]
    FFROT: U2[1]
    ORLOC: B[1]
    RPSEL: U1[1]
    REF: [
        REFP: I4[2]
        ...
    ]
    DUTMS: A[n]
    XDIES: U4[1]
    YDIES: U4[1]
    ROWCT: U4[1]
```

```
        COLCT: U4[1]
        NULBC: A[n]
        PRDCT: U4[1]
        PRAXI: B[1]
    }
```

Example:

```
    >>> import secsgem
    >>> secsgem.SecsS12F01({"MID": "materialID",
            "IDTYP": 0,
            "FNLOC": 0,
            "FFROT": 0,
            "ORLOC": 0,
            "RPSEL": 0,
            "REF": [[1,2], [2,3]],
            "DUTMS": "unit",
            "XDIES": 100,
            "YDIES": 100,
            "ROWCT": 10,
            "COLCT": 10,
            "NULBC": "{x}",
            "PRDCT": 100,
            "PRAXI": 0,
            })
    S12F1 { [MID: A 'materialID', IDTYP: B 0, FNLOC: U2 0, FFROT: U2 0, ORLOC: B 0, RPSEL: U1 0, REF
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F02**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 02 - map setup data - acknowledge

Structure:

```
    SDACK: B[1]
```

Example:

```
    >>> import secsgem
    >>> secsgem.SecsS12F02(0)
    S12F2 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F03**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 03 - map setup data - request

Structure:

```
    {
        MID: A[16]
        IDTYP: B[1]
        MAPFT: B[1]
        FNLOC: U2[1]
        FFROT: U2[1]
```

```
        ORLOC: B[1]
        PRAXI: B[1]
        BCEQU: U1[n]
        NULBC: A[n]
    }
```

Example:

```
    >>> import secsgem
    >>> secsgem.SecsS12F03({"MID": "materialID",
            "IDTYP": 0,
            "MAPFT": 0,
            "FNLOC": 0,
            "FFROT": 0,
            "ORLOC": 0,
            "PRAXI": 0,
            "BCEQU": [1, 3, 5, 7],
            "NULBC": "{x}",
            })
    S12F3 { [MID: A 'materialID', IDTYP: B 0, MAPFT: B 0, FNLOC: U2 0, FFROT: U2 0, ORLOC: B 0, PRAX
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F04**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 04 - map setup data

Structure:

```
    {
        MID: A[16]
        IDTYP: B[1]
        FNLOC: U2[1]
        ORLOC: B[1]
        RPSEL: U1[1]
        REF: [
            REFP: I4[2]
            ...
        ]
        DUTMS: A[n]
        XDIES: U4[1]
        YDIES: U4[1]
        ROWCT: U4[1]
        COLCT: U4[1]
        PRDCT: U4[1]
        BCEQU: U1[n]
        NULBC: A[n]
        MLCL: U4[1]
    }
```

Example:

```
    >>> import secsgem
    >>> secsgem.SecsS12F04({"MID": "materialID",
            "IDTYP": 0,
            "FNLOC": 0,
            "ORLOC": 0,
            "RPSEL": 0,
```

```
            "REF": [[1,2], [2,3]],
            "DUTMS": "unit",
            "XDIES": 100,
            "YDIES": 100,
            "ROWCT": 10,
            "COLCT": 10,
            "PRDCT": 100,
            "BCEQU": [1, 3, 5, 7],
            "NULBC": "{x}",
            "MLCL": 0,
            })
    S12F4 { [MID: A 'materialID', IDTYP: B 0, FNLOC: U2 0, ORLOC: B 0, RPSEL: U1 0, REF: [I4 [1, 2],
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F05**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 05 - map transmit inquire

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    MAPFT: B[1]
    MLCL: U4[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F05({"MID": "materialID", "IDTYP": 0, "MAPFT": 0, "MLCL": 0})
S12F5 { [MID: A 'materialID', IDTYP: B 0, MAPFT: B 0, MLCL: U4 0] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F06**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 06 - map transmit - grant

Structure:

```
GRNT1: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F06(0)
S12F6 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F07**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 07 - map data type 1 - send

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    [
        {
            RSINF: I4[3]
            BINLT: U1[]
        }
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F07({"MID": "materialID", "IDTYP": 0, "DATA": [{"RSINF": [1, 2, 3], "BINLT":
S12F7 { [MID: A 'materialID', IDTYP: B 0, DATA: [[RSINF: I4 [1, 2, 3], BINLT: U1 [1, 2, 3, 4]],
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F08**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 08 - map data type 1 - acknowledge

Structure:

```
MDACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F08(0)
S12F8 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F09**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 09 - map data type 2 - send

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    STRP: I2[2]
    BINLT: U1[]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F09({"MID": "materialID", "IDTYP": 0, "STRP": [0, 1], "BINLT": [1, 2, 3, 4, 5
S12F9 { [MID: A 'materialID', IDTYP: B 0, STRP: I2 [0, 1], BINLT: U2 [1, 2, 3, 4, 5, 6]] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F10**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 10 - map data type 2 - acknowledge

Structure:

```
MDACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F10(0)
S12F10 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F11**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 11 - map data type 3 - send

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    [
        {
            XYPOS: I2[2]
            BINLT: U1[]
        }
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F11({"MID": "materialID", "IDTYP": 0, "DATA": [{"XYPOS": [1, 2], "BINLT": [1,
S12F11 { [MID: A 'materialID', IDTYP: B 0, DATA: [[XYPOS: I2 [1, 2], BINLT: U1 [1, 2, 3, 4]], [X
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F12**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 12 - map data type 3 - acknowledge

Structure:

```
MDACK: B[1]
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F12(0)
S12F12 { B 0 }
```

Parameters **value** (*byte*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F13**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 13 - map data type 1 - request

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F13({"MID": "materialID", "IDTYP": 0})
S12F13 { [MID: A 'materialID', IDTYP: B 0] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F14**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 14 - map data type 1

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    [
        {
            RSINF: I4[3]
            BINLT: U1[]
        }
    ]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F14({"MID": "materialID", "IDTYP": 0, "DATA": [{"RSINF": [1, 2, 3], "BINLT":
S12F14 { [MID: A 'materialID', IDTYP: B 0, DATA: [[RSINF: I4 [1, 2, 3], BINLT: U1 [1, 2, 3, 4]],
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F15**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 15 - map data type 2 - request

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F15({"MID": "materialID", "IDTYP": 0})
S12F15 { [MID: A 'materialID', IDTYP: B 0] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F16**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 16 - map data type 2

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    STRP: I2[2]
    BINLT: U1[]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F16({"MID": "materialID", "IDTYP": 0, "STRP": [0, 1], "BINLT": [1, 2, 3, 4, 5
S12F16 { [MID: A 'materialID', IDTYP: B 0, STRP: I2 [0, 1], BINLT: U2 [1, 2, 3, 4, 5, 6]] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F17**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 17 - map data type 3 - request

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
    SDBIN: B[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F17({"MID": "materialID", "IDTYP": 0, "SDBIN": 1})
S12F17 { [MID: A 'materialID', IDTYP: B 0, SDBIN: B 1] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F18**(*value=None*)
Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 18 - map data type 3

Structure:

```
{
    MID: A[16]
    IDTYP: B[1]
```

```
        [
            {
                XYPOS: I2[2]
                BINLT: U1[]
            }
        ]
    }
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F18({"MID": "materialID", "IDTYP": 0, "DATA": [{"XYPOS": [1, 2], "BINLT": [1,
S12F18 { [MID: A 'materialID', IDTYP: B 0, DATA: [[XYPOS: I2 [1, 2], BINLT: U1 [1, 2, 3, 4]], [X
```

Parameters **value** (*dict*) – parameters for this function (see example)

class secsgem.secs.functions.**SecsS12F19**(*value=None*)

Bases: *secsgem.secs.functionbase.SecsStreamFunction*

Secs stream and function class for stream 12, function 19 - map error report - send

Structure:

```
{
    MAPER: B[1]
    DATLC: U1[1]
}
```

Example:

```
>>> import secsgem
>>> secsgem.SecsS12F19({"MAPER": 1, "DATLC": 0})
S12F19 { [MAPER: B 1, DATLC: U1 0] }
```

Parameters **value** (*dict*) – parameters for this function (see example)

Handler

Handler for SECS commands. Used in combination with secsgem.HsmsHandler.HsmsConnectionManager

class secsgem.secs.handler.**SecsHandler**(*address*, *port*, *active*, *session_id*, *name*, *event_handler=None*, *custom_connection_handler=None*)

Bases: *secsgem.common.StreamFunctionCallbackHandler*, *secsgem.hsms.handler.HsmsHandler*

Baseclass for creating Host/Equipment models. This layer contains the SECS functionality. Inherit from this class and override required functions.

Parameters

- **address** (*string*) – IP address of remote host
- **port** (*integer*) – TCP port of remote host
- **active** (*boolean*) – Is the connection active (*True*) or passive (*False*)
- **session_id** (*integer*) – session / device ID to use for connection
- **name** (*string*) – Name of the underlying configuration

- **event_handler** (*secsgem.common.EventHandler*) – object for event handling
- **custom_connection_handler** (*secsgem.hsms.connections.HsmsMultiPassiveServer*) – object for connection handling (ie multi server)

ceids = {}
Dictionary of available collection events, CEID is the key

Parameters

- **name** (*string*) – Name of the data value
- **CEID** (*integer*) – Collection event the data value is used for

dvs = {}
Dictionary of available data values, DVID is the key

Parameters

- **name** (*string*) – Name of the collection event
- **dv** (*list of integers*) – Data values available for collection event

alarms = {}
Dictionary of available alarms, ALID is the key

Parameters

- **alarmText** (*string*) – Description of the alarm
- **ceidOn** (*integer*) – Collection event for activated alarm
- **ceidOff** (*integer*) – Collection event for deactivated alarm

rcmds = {}
Dictionary of available remote commands, command is the key

Parameters

- **params** (*list of dictionary*) – description of the parameters
- **CEID** (*list of integers*) – Collection events the remote command uses

secsStreamsFunctionsHost = {0: {0: <class 'secsgem.secs.functions.SecsS00F00'>}, 1: {0: <class 'secsgem.secs.funct

secsStreamsFunctionsEquipment = {0: {0: <class 'secsgem.secs.functions.SecsS00F00'>}, 1: {0: <class 'secsgem.se

disable_ceids()
Disable all Collection Events.

disable_ceid_reports()
Disable all Collection Event Reports.

list_svs()
Get list of available Service Variables.

Returns available Service Variables

Return type list

request_svs(*svs*)
Request contents of supplied Service Variables.

Parameters **svs** (*list*) – Service Variables to request

Returns values of requested Service Variables

Return type list

request_sv(*sv*)
Request contents of one Service Variable.

Parameters **sv** (*int*) – id of Service Variable

Returns value of requested Service Variable

Return type various

list_ecs()
Get list of available Equipment Constants.

Returns available Equipment Constants

Return type list

request_ecs(*ecs*)
Request contents of supplied Equipment Constants.

Parameters **ecs** (*list*) – Equipment Constants to request

Returns values of requested Equipment Constants

Return type list

request_ec(*ec*)
Request contents of one Equipment Constant.

Parameters **ec** (*int*) – id of Equipment Constant

Returns value of requested Equipment Constant

Return type various

set_ecs(*ecs*)
Set contents of supplied Equipment Constants.

Parameters **ecs** (*list*) – list containing list of id / value pairs

set_ec(*ec*, *value*)
Set contents of one Equipment Constant.

Parameters

- **ec** (*int*) – id of Equipment Constant
- **value** (*various*) – new content of Equipment Constant

send_equipment_terminal(*terminal_id*, *text*)
Set text to equipment terminal

Parameters

- **terminal_id** (*int*) – ID of terminal
- **text** (*string*) – text to send

get_ceid_name(*ceid*)
Get the name of a collection event

Parameters **ceid** (*integer*) – ID of collection event

Returns Name of the event or empty string if not found

Return type string

disable()
Disables the connection

enable()
Enables the connection

fire_event(*event_name*, *data*, *async=False*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **data** (*dict*) – parameters for event

get_dvid_name(*dvid*)
Get the name of a data value

Parameters **dvid** (*integer*) – ID of data value

Returns Name of the event or empty string if not found

Return type string

on_connection_before_closed(_)
Connection is about to be closed

on_connection_closed(_)
Connection was closed

on_connection_established(_)
Connection was established

on_connection_packet_received(_, *packet*)
Packet received by connection

Parameters **packet** (*secsgem.hsms.packets.HsmsPacket*) – received data packet

register_callback(*stream*, *function*, *callback*)
Register the function callback for stream and function. Multiple callbacks can be registered for one function.

Parameters

- **stream** (*integer*) – stream to register callback for
- **function** (*integer*) – function to register callback for
- **callback** (*def callback(connection)*) – method to call when stream and functions is received

send_and_waitfor_response(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_deselect_req()
Send a Deselect Request to the remote host

Returns System of the sent request

Return type integer

send_deselect_rsp(*system_id*)
Send a Deselect Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_linktest_req()
Send a Linktest Request to the remote host

Returns System of the sent request

Return type integer

send_linktest_rsp(*system_id*)
Send a Linktest Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_reject_rsp(*system_id*, *s_type*, *reason*)
Send a Reject Response to the remote host

Parameters

- **system_id** (*integer*) – System of the request to reply for
- **s_type** (*integer*) – s_type of rejected message
- **reason** (*integer*) – reason for rejection

send_response(*function*, *system*)
Send response function for system

Parameters

- **function** (*secsgem.secs.functionbase.SecsStreamFunction*) – function to be sent
- **system** (*integer*) – system to reply to

send_select_req()
Send a Select Request to the remote host

Returns System of the sent request

Return type integer

send_select_rsp(*system_id*)
Send a Select Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_separate_req()
Send a Separate Request to the remote host

send_stream_function(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

unregister_callback(*stream*, *function*, *callback*)
Unregister the function callback for stream and function. Multiple callbacks can be registered for one function, only the supplied callback will be removed.

Parameters

- **stream** (*integer*) – stream to unregister callback for
- **function** (*integer*) – function to register callback for
- **callback** (*def callback(connection)*) – method to remove from callback list

waitfor_deselect_rsp(*system_id*)
Wait for an incoming Deselect Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_linktest_rsp(*system_id*)
Wait for an incoming Linktest Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_select_rsp(*system_id*)
Wait for an incoming Select Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_stream_function(*stream*, *function*, *is_control=False*)
Wait for an incoming stream and function and return the receive data

Parameters

- **stream** (*integer*) – number of stream to wait for
- **function** (*integer*) – number of function to wait for
- **is_control** (*bool*) – is it a control packet

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_system(*system*, *is_control=False*)
Wait for an message with supplied system

Parameters **system** (*integer*) – number of system to wait for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

are_you_there()
Check if remote is still replying

stream_function(*stream*, *function*)
Get class for stream and function

Parameters

- **stream** (*int*) – stream to get function for
- **function** (*int*) – function to get

Returns matching stream and function class

Return type secsSxFx class

secs_decode(*packet*)
Get object of decoded stream and function class, or None if no class is available.

Parameters **packet** (*secsgem.hsms.packets.HsmsPacket*) – packet to get object for

Returns matching stream and function object

Return type secsSxFx object

3.4.3 GEM

Handler

Handler for GEM commands. Used in combination with `secsgem.HsmsHandler.HsmsConnectionManager`

class `secsgem.gem.handler.`**GemHandler**(*address*, *port*, *active*, *session_id*, *name*, *event_handler=None*, *custom_connection_handler=None*)

Bases: *secsgem.secs.handler.SecsHandler*

Baseclass for creating Host/Equipment models. This layer contains GEM functionality. Inherit from this class and override required functions.

Parameters

- **address** (*string*) – IP address of remote host
- **port** (*integer*) – TCP port of remote host
- **active** (*boolean*) – Is the connection active (*True*) or passive (*False*)
- **session_id** (*integer*) – session / device ID to use for connection
- **name** (*string*) – Name of the underlying configuration
- **event_handler** (*secsgem.common.EventHandler*) – object for event handling
- **custom_connection_handler** (*secsgem.hsms.connections.HsmsMultiPassiveServer*) – object for connection handling (ie multi server)

ceids = {}

Dictionary of available collection events, CEID is the key

Parameters

- **name** (*string*) – Name of the data value
- **CEID** (*integer*) – Collection event the data value is used for

dvs = {}

Dictionary of available data values, DVID is the key

Parameters

- **name** (*string*) – Name of the collection event
- **dv** (*list of integers*) – Data values available for collection event

alarms = {}

Dictionary of available alarms, ALID is the key

Parameters

- **alarmText** (*string*) – Description of the alarm
- **ceidOn** (*integer*) – Collection event for activated alarm
- **ceidOff** (*integer*) – Collection event for deactivated alarm

rcmds = {}
Dictionary of available remote commands, command is the key

Parameters

- **params** (*list of dictionary*) – description of the parameters
- **CEID** (*list of integers*) – Collection events the remote command uses

enable()
Enables the connection

disable()
Disables the connection

on_connection_closed(*connection*)
Connection was closed

clear_collection_events()
Clear all collection events

subscribe_collection_event(*ceid*, *dvs*, *report_id=None*)
Subscribe to a collection event

Parameters

- **ceid** (*integer*) – ID of the collection event
- **dvs** (*list of integers*) – DV IDs to add for collection event
- **report_id** (*integer*) – optional - ID for report, autonumbering if None

send_remote_command(*rcmd*, *params*)
Send a remote command

Parameters

- **rcmd** (*string*) – Name of command
- **params** (*list of strings*) – DV IDs to add for collection event

send_process_program(*ppid*, *ppbody*)
Send a process program

Parameters

- **ppid** (*string*) – Transferred process programs ID
- **ppbody** (*string*) – Content of process program

request_process_program(*ppid*)
Request a process program

Parameters **ppid** (*string*) – Transferred process programs ID

delete_process_programs(*ppids*)
Delete a list of process program

Parameters **ppids** (*list of strings*) – Process programs to delete

get_process_program_list()
Get process program list

are_you_there()
Check if remote is still replying

disable_ceid_reports()
Disable all Collection Event Reports.

disable_ceids()
Disable all Collection Events.

fire_event(*event_name*, *data*, *async=False*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **data** (*dict*) – parameters for event

get_ceid_name(*ceid*)
Get the name of a collection event

Parameters **ceid** (*integer*) – ID of collection event

Returns Name of the event or empty string if not found

Return type string

get_dvid_name(*dvid*)
Get the name of a data value

Parameters **dvid** (*integer*) – ID of data value

Returns Name of the event or empty string if not found

Return type string

list_ecs()
Get list of available Equipment Constants.

Returns available Equipment Constants

Return type list

list_svs()
Get list of available Service Variables.

Returns available Service Variables

Return type list

on_connection_before_closed(*_*)
Connection is about to be closed

on_connection_established(*_*)
Connection was established

on_connection_packet_received(*_*, *packet*)
Packet received by connection

Parameters **packet** (*secsgem.hsms.packets.HsmsPacket*) – received data packet

register_callback(*stream*, *function*, *callback*)
Register the function callback for stream and function. Multiple callbacks can be registered for one function.

Parameters

- **stream** (*integer*) – stream to register callback for
- **function** (*integer*) – function to register callback for

- **callback** (*def callback(connection)*) – method to call when stream and functions is received

request_ec(*ec*)
Request contents of one Equipment Constant.

Parameters **ec** (*int*) – id of Equipment Constant

Returns value of requested Equipment Constant

Return type various

request_ecs(*ecs*)
Request contents of supplied Equipment Constants.

Parameters **ecs** (*list*) – Equipment Constants to request

Returns values of requested Equipment Constants

Return type list

request_sv(*sv*)
Request contents of one Service Variable.

Parameters **sv** (*int*) – id of Service Variable

Returns value of requested Service Variable

Return type various

request_svs(*svs*)
Request contents of supplied Service Variables.

Parameters **svs** (*list*) – Service Variables to request

Returns values of requested Service Variables

Return type list

s01f01_handler(*handler*, *packet*)
Callback handler for Stream 1, Function 1, Are You There

See also:

secsgem.common.StreamFunctionCallbackHandler.register_callback()

Parameters

- **handler** (*secsgem.hsms.handler.HsmsHandler*) – handler the message was received on
- **packet** (*secsgem.hsms.packets.HsmsPacket*) – complete message received

secsStreamsFunctionsEquipment = {0: {0: <class 'secsgem.secs.functions.SecsS00F00'>}, 1: {0: <class 'secsgem.se

secsStreamsFunctionsHost = {0: {0: <class 'secsgem.secs.functions.SecsS00F00'>}, 1: {0: <class 'secsgem.secs.funct

secs_decode(*packet*)
Get object of decoded stream and function class, or None if no class is available.

Parameters **packet** (*secsgem.hsms.packets.HsmsPacket*) – packet to get object for

Returns matching stream and function object

Return type secsSxFx object

send_and_waitfor_response(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

send_deselect_req()
Send a Deselect Request to the remote host

Returns System of the sent request

Return type integer

send_deselect_rsp(*system_id*)
Send a Deselect Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_equipment_terminal(*terminal_id*, *text*)
Set text to equipment terminal

Parameters

- **terminal_id** (*int*) – ID of terminal
- **text** (*string*) – text to send

send_linktest_req()
Send a Linktest Request to the remote host

Returns System of the sent request

Return type integer

send_linktest_rsp(*system_id*)
Send a Linktest Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_reject_rsp(*system_id*, *s_type*, *reason*)
Send a Reject Response to the remote host

Parameters

- **system_id** (*integer*) – System of the request to reply for
- **s_type** (*integer*) – s_type of rejected message
- **reason** (*integer*) – reason for rejection

send_response(*function*, *system*)
Send response function for system

Parameters

- **function** (*secsgem.secs.functionbase.SecsStreamFunction*) – function to be sent
- **system** (*integer*) – system to reply to

send_select_req()
Send a Select Request to the remote host

Returns System of the sent request

Return type integer

send_select_rsp(*system_id*)
Send a Select Response to the remote host

Parameters **system_id** (*integer*) – System of the request to reply for

send_separate_req()
Send a Separate Request to the remote host

send_stream_function(*packet*)
Send the packet and wait for the response

Parameters **packet** (*secsgem.secs.functionbase.SecsStreamFunction*) – packet to be sent

set_ec(*ec*, *value*)
Set contents of one Equipment Constant.

Parameters

- **ec** (*int*) – id of Equipment Constant
- **value** (*various*) – new content of Equipment Constant

set_ecs(*ecs*)
Set contents of supplied Equipment Constants.

Parameters **ecs** (*list*) – list containing list of id / value pairs

stream_function(*stream*, *function*)
Get class for stream and function

Parameters

- **stream** (*int*) – stream to get function for
- **function** (*int*) – function to get

Returns matching stream and function class

Return type secsSxFx class

unregister_callback(*stream*, *function*, *callback*)
Unregister the function callback for stream and function. Multiple callbacks can be registered for one function, only the supplied callback will be removed.

Parameters

- **stream** (*integer*) – stream to unregister callback for
- **function** (*integer*) – function to register callback for
- **callback** (*def callback(connection)*) – method to remove from callback list

waitfor_deselect_rsp(*system_id*)
Wait for an incoming Deselect Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_linktest_rsp(*system_id*)
Wait for an incoming Linktest Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_select_rsp (*system_id*)
Wait for an incoming Select Response

Parameters **system_id** (*integer*) – System of the request to reply for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_stream_function (*stream*, *function*, *is_control=False*)
Wait for an incoming stream and function and return the receive data

Parameters

- **stream** (*integer*) – number of stream to wait for
- **function** (*integer*) – number of function to wait for
- **is_control** (*bool*) – is it a control packet

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

waitfor_system (*system*, *is_control=False*)
Wait for an message with supplied system

Parameters **system** (*integer*) – number of system to wait for

Returns Packet that was received

Return type *secsgem.hsms.packets.HsmsPacket*

s01f13_handler (*handler*, *packet*)
Callback handler for Stream 1, Function 13, Establish Communication Request

See also:

secsgem.common.StreamFunctionCallbackHandler.register_callback()

Parameters

- **handler** (*secsgem.hsms.handler.HsmsHandler*) – handler the message was received on
- **packet** (*secsgem.hsms.packets.HsmsPacket*) – complete message received

s06f11_handler (*handler*, *packet*)
Callback handler for Stream 6, Function 11, Establish Communication Request

See also:

secsgem.common.StreamFunctionCallbackHandler.register_callback()

Parameters

- **handler** (*secsgem.hsms.handler.HsmsHandler*) – handler the message was received on
- **packet** (*secsgem.hsms.packets.HsmsPacket*) – complete message received

s10f01_handler(*handler*, *packet*)
Callback handler for Stream 10, Function 1, Terminal Request

See also:

secsgem.common.StreamFunctionCallbackHandler.register_callback()

Parameters

- **handler** (*secsgem.hsms.handler.HsmsHandler*) – handler the message was received on
- **packet** (*secsgem.hsms.packets.HsmsPacket*) – complete message received

3.4.4 Common functionality

Contains helper functions

secsgem.common.**format_hex**(*text*)
Returns byte arrays (string) formated as hex numbers.

Example:

```
>>> import secsgem
>>>
>>> data = "asdfg"
>>> secsgem.common.format_hex(data)
'61:73:64:66:67'
```

Parameters **text** (*string*) – byte array

Returns Formated text

Return type string

secsgem.common.**is_windows**()
Returns True if running on windows

Returns Is windows system

Return type bool

secsgem.common.**function_name**(*function*)
Gets name of function or method

Returns function/method name

Return type string

class secsgem.common.**StreamFunctionCallbackHandler**
Base class for all connection classes. Provides functionality for registering and unregistering callbacks for streams and functions.

register_callback(*stream*, *function*, *callback*)
Register the function callback for stream and function. Multiple callbacks can be registered for one function.

Parameters

- **stream** (*integer*) – stream to register callback for
- **function** (*integer*) – function to register callback for

- **callback** (*def callback(connection)*) – method to call when stream and functions is received

unregister_callback(*stream*, *function*, *callback*)
Unregister the function callback for stream and function. Multiple callbacks can be registered for one function, only the supplied callback will be removed.

Parameters

- **stream** (*integer*) – stream to unregister callback for
- **function** (*integer*) – function to register callback for
- **callback** (*def callback(connection)*) – method to remove from callback list

class secsgem.common.**EventHandler**(*target=None*, *events=None*, *generic_handler=None*)
Class for event handling. Provides functionality for managing events.

Parameters

- **target** (*object*) – receiver object for event callbacks
- **generic_handler** (*def handler(eventName, data)*) – receiver function for all events

Params events dictionary of event names with handlers

fire_event(*event_name*, *params*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **params** (*dict*) – parameters for event

add_event_handler(*event_name*, *handler*)
Register handler for an event. Multiple handlers can be registered for one event.

Parameters

- **event_name** (*string*) – event to register handler for
- **handler** (*def handler(event_name, handler)*) – method to call when event is received

remove_event_handler(*event_name*, *handler*)
Unregister handler for an event.

Parameters

- **event_name** (*string*) – event to unregister handler for
- **handler** (*def handler(event_name, handler)*) – method to unregister

class secsgem.common.**EventProducer**(*event_handler*)
Class for event production. Provides functionality for sending events.

Parameters **event_handler** (*secsgem.common.EventHandler*) – object for event handling

fire_event(*event_name*, *data*, *async=False*)
Fire an event

Parameters

- **event_name** (*string*) – event to fire
- **data** (*dict*) – parameters for event

- genindex
- modindex
- search

S

A

C

D

E

F

G

H

I

L

O

R

S

T

U

W