API reference- gspentest.iec61850.goose

For importing to python code: **Import gspentest.iec61850**

# Class: gspentest.iec61850.goose

# Nested classes:

# Private method:

|  |  |
| --- | --- |
| **Function name** | **Details** |
| mac\_to\_hex(mac\_id) | Convert mac id it to standard format 0xffffffffffff |

mac\_to\_hex()

|  |  |
| --- | --- |
| **Parameters:** | **mac\_id : string**  Specifies the the mac id in formats such as ff-ff-ff-ff-ff-ff, ff:ff:ff:ff:ff:ff, ff-ff-ff-ff-ff-ff/ ff-ff-ff-ff-ff-ff or ff:ff:ff:ff:ff:ff/ ff:ff:ff:ff:ff:ff |
| **Returns**: | Mac\_id: hex  Returns a hexadecimal number with the format 0xffffffffffff |
| **Usage:** | out =mac\_to\_hex(“ff:ff:ff:ff:ff:ff”)  print(out)  >> 0xffffffffffff |

# Functions:

|  |  |
| --- | --- |
| **Function name** | **Details** |
| goose.snoop(interface) | Capture goose packets |
| goose.filter(packet) | Filter goose packets based on specified filters |
| goose.modify() | Modify different aspects of the packet |
| goose.push() | Push the packet on the specified networking interface with time delay between packets |
| goose.report() | Generate report for the goose packets received |

**goose.snoop**(interface, t=1000)

|  |  |
| --- | --- |
| **Parameters:** | **interface : string**  Specifies the network interface to start capturing messages on.  **t : int, default= 1000**  Time for which the packets are to be captured. Default 1 seconds |
| **Returns**: | packet: dict  Returns a dictionary containing all the GOOSE packets |
| **Usage:** | out =goose.snoop(“eth0”, t=15000)  print(out)  >> {timestamp:[1.1, 1.2], length:[167,167], } |

**goose.filter**(packet, src\_mac=0x000000000000/0x000000000000, dst\_mac=0x000000000000/0x000000000000, length=-1, capture\_time=[0, -1])

|  |  |
| --- | --- |
| **Parameters:** | **packet : dict**  The dictionary containg all goose packets on which the filter is to be applied  **src\_mac : {“hex”, “hex/hex\_mask”, [hex, hex, hex,..]},** default=0x000000000000/0x000000000000  Filter based on source mac id. Input can be single mac id(string), mac id with mask(string) and an array of mac ids(array of strings). Default value is all possible mac ids  **dst\_mac : {“hex”, “hex/hex\_mask”, [hex, hex, hex,..]},** default=0x000000000000/0x000000000000  Filter based on destination mac id. Input can be single mac id (string), mac id with mask(string) and an array of mac ids(array of strings). Default value is all possible mac ids  **length : {int, [int, int]}, default=-1**  Filter based on length of packet. Input can be integer, array of 2 integers. If an array is given, the first element corresponds to lower limit of length and the second corresponds to upper limit of length. Default value is -1 which specifies that all packets need to be selected.  **capture\_time : {“float”, [“float”, “float”]}, default = [0,-1]**  Filter base on capture time. Input can be float where one time is specified or array of 2 floats. If an array is given, the first element corresponds to lower limit of captured time and the second corresponds to upper limit of captured time. |
| **Returns**: | **packet : dict** |
| **Usage:** | Source mac format: 11:ff:06:30:01:01/11-ff-06-30-01-01/0x11ff06300101  **goose.filter**(packet, src\_mac=0x11ff06300101/0xffff000000000, length=167, capture\_time=[0, -1]) |

**goose.modify**(packet, gocb=None, payload=None, stnum=None, sqnum=None,capture\_time=None, src\_mac=None, dst\_mac=None, appid=None, length=none)

|  |  |
| --- | --- |
| **Parameters**: | **packet : dict**  Input is a dict type variable which containing one or multiple rows.  **gocb : {str, [“str”, …]}, default = None**  Modify the gocb value in the packet. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **Payload : array, default= None**  The originally Payload is removed and this payload is attached. The size of old and new payload need not be same. If a single packet is to be modified then 1-d array is the input. If multiple packets need to be modified, then 2-D array is the input where number of elements matching the number of rows in dict. No change if not specified  **stnum : {int, [int, …]}, default= None**  Modify the stnum parameter. If a single packet is to be modified then int is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **sqnum : {int, [int, …]}, default= None**  Modify the sqnum parameter. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **time : {float, [float, …]}, default = None**  Modify the timestamp in the packet. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **src\_mac= {hex, [hex, …]}, default= None**  Modify the source mac id. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **dst\_mac= {hex, [hex, …]}, default= None**  Modify the destination mac id. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **appid: {int, [int, …]} default= None**  Modify the appid of the packet. If a single packet is to be modified then string is the input. If multiple packets need to be modified, then array is the input where number of elements matching the number of rows in dict. No change if not specified  **length: {int, “nochange”, [int, …]}, default = None**  Custom change the length of the packet by specifying the integer length or enter string “no change” tokeep the original length. If not specified, the code automatically calculates the correct length of the packet and modifies it. If input packet has multiple rows, then array input is given. |
| **Returns**: | **Packet:dict** |
| **Usage:** | **goose.modify**(packet, gocb=IED3simpleIO$meas$GGIO.3, payload=[[True, True]], stnum=8, sqnum=150)  or  **goose.modify**(packet, gocb=IED3simpleIO$meas$GGIO.3, payload=[[True, True],[False, True],[False False]], stnum=[7,8,9] sqnum=[150,1,1]) |

**goose.push**(packet, interface, delay = 100)

|  |  |
| --- | --- |
| **Parameters**: | **packet: dict**  The dictionary of packet needed to be pushed in the network.  **Interface: str**  Networking interface to be used for pushing the packets  **Delay: {int, array}, default = 100ms**  Delay between the consecutive packets. The integer value in milliseconds is to be specified for constant time between the packets. An array of any size can be specified. The function will create a repeated array of size same as packet dict based on the delay array. |
| **Returns**: | **None** |
| **Usage:** | **goose.push**(packet, “eth1”, delay = 10)  **goose.push**(packet, “eth1”, delay = [10, 15, 20]) |