API Documentation

API Documentation

November 8, 2017

Contents

1 Package ethanol

```
This package contains some components to implement Ethanol API.
ethanol should run as a pox module

sample command call:
    python ./pox.py forwarding.l2_learning ethanol.server

ethanol.server is the ~/ethanol/python/server.py file

you must create a symbolic link inside pox subtree, like:
cd ~/ethanol/pox/pox
ln ~/ethanol/python ethanol
```

1.1 Modules

  (Section 46, p. 104)
  - **msg_mean_sta_stats**: implements the following messages:
    (Section 47, p. 105)
  - **msg_memcpu**: implements the following messages:
    (Section 48, p. 109)
  - **msg_mtu_qlen**: implements: * set_txqueuelen * set_mtu
    (Section 49, p. 111)
  - **msg_ping**: implements:
    (Section 50, p. 113)
  - **msg_powersave**: implements the following messages:
    (Section 51, p. 115)
  - **msg_preamble**: implements: * get_preamble * set_preamble
    (Section 52, p. 117)
  - **msg_radio_wlans**: implements the following messages:
    (Section 53, p. 119)
  - **msg_sent_received**: implements the following messages:
    (Section 54, p. 121)
  - **msg_server**: this is creates the server, that deals with clients (aps and stations) messages the messages implemented are mapped in map_msg_to_procedure main entry to this module is: call run(server)
    (Section 55, p. 126)
  - **msg_snr_power**: implements the following messages:
    (Section 56, p. 128)
  - **msg_ssid**: implements the following messages:
    (Section 57, p. 131)
  - **msg_sta_link_information**: implements the following messages:
    (Section 58, p. 133)
  - **msg_sta_statistics**: implements the following messages:
    (Section 59, p. 135)
  - **msg_station_trigger_transition**: implements the following messages:
    (Section 60, p. 137)
  - **msg_statistics**: implements the following messages:
    (Section 61, p. 138)
  - **msg_tos**: implements the following messages:
    (Section 62, p. 140)
  - **msg_uptime**: implements the following messages:
    (Section 63, p. 142)
  - **msg_wlan_info**: implements: * req_wlan_info(): MSG_WLAN_INFO
    (Section 64, p. 143)
- **tos**: This package contains some components to implement Ethanol API.
  (Section 65, p. 145)
  - **usecase_tos**: This is a module that runs inside POX.
    (Section 66, p. 146)

1.2 Variables

Name	Description
___package___	**Value:** `None`

2 Module ethanol.client__test

For TESTING purpose only. Don't use it as a template to your code. This module uses construct (https://pypi.python.org/pypi/construct) See more info at msg__core.py on how to install it.

We use this module to test ethanol messages. It does not use Ethanol architecture, only its messages. We must import the correct message module, and place its call in launch()

This is a pox module. It should by called using pox.py.

Command sample:

cd pox ./pox.py ethanol.client__test –server__address='thunder' –server__port=22223

2.1 Functions

msg__acs(*connect*, *intf__name*=`'wlan0'`, *num__acs__tests*=`1`)

this is a test function. it runs num__acs__tests times on interface wlan0

launch(*server__address*=`'0.0.0.0'`, *server__port*=`'22223'`, *num__acs__tests*=`1`, *intf__name*=`'wlan0'`, *mac__sta*=`'0c:84:dc:d4:7a:73'`)

launch is a default method used by pox to load and run this module

3 Package ethanol.ethanol

This package contains the main classes to implement Ethanol API.

See Also: file Entidades-vxxxx.pdf contains the class diagram for this API

Change Log:

- Entidades-v1.pdf
- Entidades-v2.pdf
- Entidades-v3.pdf

3.1 Modules

- **ap**: Defines the AP class.
  (Section 4, p. 7)
- **device**: This module provides: class device.Device
  (Section 5, p. 12)
- **network**: defines the Network class that represents the SSIDs controlled by the Ethanol Controller
  (Section 6, p. 16)
- **radio**: This module provides: class radio.Radio
  (Section 7, p. 19)
- **station** *(Section 8, p. 23)*
- **switch**: An L2 learning switch based on L2 learning example from POX
  (Section 9, p. 26)
- **vap**: This module provides: class VAP
  (Section 10, p. 28)

3.2 Variables

Name	Description
__package__	**Value:** `None`

4 Module ethanol.ethanol.ap

Defines the AP class. It represents the physical access point.

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

4.1 Functions

connected_aps()

use this function to get the dictionary that contains all aps currently connected to Ethanol controller

Return Value

list of ap's objects

is_ap_with_ip_connected(*ip*)

Return Value

TRUE if an AP with the ip provided as a parameter is connected

Note: this is the ip of the AP's interface that sends packets to the controller, i.e., normally it is an ethernet interface

get_ap_by_ip(*ip*)

get the AP object with an IP address (of the connection to the controller)

Parameters

`ip`: a string with the ip address in dotted format

Return Value

the AP object that has the provided ip address, or None if it doesn't exist

get_vap_by_mac_address(*mac_address*)

get a VAP object by its MAC address (BSSID)

Parameters

`mac_address`: MAC address in dotted format of the Virtual AP (SSID)

Return Value

a VAP object that matches the mac_address or None if doesn't match

add__ap__openflow(*ip*)

called at ethanol.server when connectionUp occurs. inserts an entry in map__openflow__vs__ethanol__ip with the ip detected in pox.openflow.connection. when a Hello message arrives, AP.___init___() searchs this mapping and assigns self to this entry

Parameters

`ip`: a string with the ip address in dotted format

(type=str)

add__ap(*client__address*)

Create (and return) an AP object for the the device represented by the tuple client__address. This function updates a list of these objects.

used by the Hello message's process

Parameters

`client_address`: tuple with (ip, port) used to make a socket connection to the AP

(type=tuple or list)

remove__ap__byIP(*ip*)

removes the ap from the list called by AP.___destroy___() or when the server receives a "bye message" from such AP

Parameters

`ip`: a string with the ip address in dotted format

(type=str)

4.2 Variables

Name	Description
map__openflow__vs__ethanol__ip	provides a mapping from the ap's ip address to the ap object **Value:** `{}`

4.3 Class AP

```
object ─┐
        │
   ethanol.ethanol.ap.AP
```

defines the AP class that represents the physical wifi device

4.3.1 Methods

__init__(*self*, *ip*, *port*=`SERVER_PORT`)

constructor

Parameters

`ip:` socket IP address to connect to the physical AP

`port:` socket port to connect to the physical AP

Overrides: object.__init__

---

id(*self*)

AP's unique identifier

Return Value

AP's uuid.uuid4() value

__del__(*self*)

Called when the instance is about to be destroyed. Removes this ap from the mapping

---

__str__(*self*)

string

Return Value

the ip and port of this device

Overrides: object.__str__

---

radios(*self*)

get list of AP's radios

Return Value

a list of radio objects associated with the AP

---

msg_id(*self*)

helper function: returns the next message id to be sent, and increments the message ID by 1

Return Value

id for the new message

---

vaps(*self*)

returns a list of the vaps configured in this AP

Return Value

list of VAP objects

createvirtualap_and_insert_listvap(*self*, *ssid*, *radio*, *mac_address*)

create the VAP based on ssid, radio, and mac_address inserts the vap in self.___listVAP list

Parameters

`ssid:` BSSID

(type=str)

`radio:` object RADIO attached to this AP

`mac_address:` MAC address in dotted format

(type=str)

Return Value

the vap created

destroyvirtualap(*self*, *vap*)

remove a VAP: deactivate it (remove SSID)

Parameters

`vap:` a vap object (SSID connected to this AP)

(type=vap.VAP object)

getsupportedinterfacemodes(*self*, *intf_name*)

indicates the modes supported

Return Value

a list with the supported modes: AP, Station, Mesh, IBSS

getinterferencemap(*self*, *intf_name*)

NOT IMPLEMENTED YET returns the interference map as defined in 802.11/2012

listwlan_interfaces(*self*)

wireless interfaces in this AP

Return Value

a list with the names of wireless interfaces in this AP

get_interface_stats(*self*)

get statistics for all interfaces

enable_interface_stats(*self*)

disable_interface_stats(*self*)

statistics_time(*self*, *new_time*)

Parameters

`new_time:` set the time of collection in miliseconds. send -1 to disable data collection

statistics_alpha(*self*, *alpha*)

defines alpha value for EWMA

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

4.3.2 Properties

Name	Description
Inherited from object	
___class___	

5 Module ethanol.ethanol.device

This module provides: class device.Device

It is a superclass for Station and VAP

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

5.1 Class Device

```
object ─┐
        └ ethanol.ethanol.device.Device
```

this superclass provides the attributes and methods shared by Station and VAP

5.1.1 Methods

__init__(*self*, *socket*, *intf_name*)

creates a device object (used by VAP and STATION)

Parameters

`socket:`	tuple (ip, port_num)
`intf_name:`	name of the wireless interface that this device uses

Overrides: object.__init__

id(*self*)

unique identifier (UUID) for this device

get_connection(*self*)

returns a tuple representing the socket to connection to the physical station

msg_id(*self*)

helper function: returns the next message id to be sent. increments the message ID by 1

intf_name(*self*)

wireless interface of this device (set during __init__)

mac_address(*self*)

wireless interface's MAC address

ipv4_address(*self*, *ip_conf*)

NOT IMPLEMENTED YET – function in C is ok

set IP v4 parameters: ip, netmask, gateway

ipv6_address(*self*, *ip_conf*)

NOT IMPLEMENTED YET – function in C is ok

set the device's IP address (version 6)

fastBSSTransition_compatible(*self*)

connect to ap requesting if it is "Fast BSS Transition" compatible

bytesReceived(*self*)

number of bytes received on this interface (cumulative value)

bytesSent(*self*)

number of bytes sent on this interface (cumulative value)

packetsReceived(*self*)

number of packets received on this interface (cumulative value)

packetsSent(*self*)

number of packets sent on this interface (cumulative value)

packetsLost(*self*)

number of packets lost on this interface (cumulative value)

jitter(*self*)

NOT IMPLEMENTED YET

Return Value

mean jitter measured at the wireless interface

delay(*self*)

NOT IMPLEMENTED YET

Return Value

mean delay measured at the wireless interface

retries(*self*)

NOT IMPLEMENTED YET

Return Value

number of retries at the wireless interface

failed(*self*)

NOT IMPLEMENTED YET

Return Value

total number of failures at the wireless interface

statistics(*self*)

collect some cumulative statistics – rx_packets, rx_bytes, rx_dropped, tx_packets, tx_bytes. these values are accumulate since the interface went up.

signalStrength(*self*)

NOT IMPLEMENTED YET

SNR(*self*)

retrieve current SNR

txpower(*self*, *new_value*)

set current tx power

tx_bitrate(*self*, *sta_mac*=`None`)

Return Value

the last seen tx_bitrate for a given station (in Mbps) or a list for each station connected (if sta_mac is None)

uptime(*self*)

system uptime and idle time in seconds

cpu(*self*)

physical device's CPU usage

cpu_usage(*self*)

same as cpu(). to keep model compatibility

memory(*self*)

physical device's memory usage

memory_usage(*self*)

same as memory(). to keep model compatibility

getAPsInRange(*self*)

get aps that are in range.

Note: this method is not precise, because it relies on the spare time the device has to scan all the channels

clear_mange(*self*)

add_tos(*self*, *rules*)

replace_tos(*self*, *rules*)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

5.1.2 Properties

Name	Description
Inherited from object	
__class__	

6 Module ethanol.ethanol.network

defines the Network class that represents the SSIDs controlled by the Ethanol Controller

This module provides:

1) add_network(net)

2) del_network(net)

3) get_or_create_network_by_ssid(ssid)

4) class Network

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

6.1 Functions

list_of_networks()

add_network(*ssid*, *net*)

returns True if successfully added the network to the set. False if the SSID of the network provided already exists. net is also not added to the set @return boolean

del_network(*net*)

delete this network: disconfigures all vaps associated to this network @param net the network to be deleted

get_or_create_network_by_ssid(*ssid*)

@return a Network object representing the ssid. if none exists, a new one is created

6.2 Class Network

object ─┐
ethanol.ethanol.network.Network

handle a network - a network is a set of VAPs that share the same SSID

6.2.1 Methods

__init__(*self*, *ssid*)

create a network with ESSID = ssid

Overrides: object.__init__

__del__(*self*)

class destructor Called when the instance is about to be destroyed.

releaseResources(*self*)

deconfigure vap's SSID

id(*self*)

returns the network's internal class ID

vaps(*self*)

returns VAPs associated to this network

SSID(*self*, *newSSID*, *keepenabled*=`False`)

change the SSID of the network

associateVirtualAP(*self*, *vap*)

join the vap to the network. called by ssid.setter in VAP class

deassociateVirtualAP(*self*, *vap*)

releases the vap from the network called by ssid.setter in VAP class

handoffUser(*station*, *new_vap*)

handles handoff. This method relies on 802.11 mobility domain feature. So the station and the AP should be configure to use mobility domain. This method disassociates the station from a vap in the network and moves it to a new_vap in this network. It also sends a message to the station, using station.triggerTransition(), instructing it to roam to a new ap.

See Also: documentacao-para-handover.pdf for instruction on how to set up the station and the AP for handover. **** not implemented yet ****

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

6.2.2 Properties

Name	Description
Inherited from object ___class___	

7 Module ethanol.ethanol.radio

This module provides: class radio.Radio

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

7.1 Class Radio

```
object ─┐
        │
    ethanol.ethanol.radio.Radio
```

Radio represents the physical radios attached to an AP

abstracts the physical radio

7.1.1 Methods

__init__(*self*, *ap*, *wiphy_name*, *ip*, *port*)

creates an object associated with the "ap" must provide the wiphy_name (intf_name)

Overrides: object.__init__

id(*self*)

Radio UUID

__str__(*self*)

returns the ip and port of this device

Overrides: object.__str__

msg_id(*self*)

handles the radio message id's

Return Value

an id to be used in the message and increments the current id

wiphy(*self*)

Return Value

the wireless interface name

validChannels(*self*)

informs a list of valid channel numbers, supported by the device in its wireless interface

Return Value

the list of the channels that can be assigned to this interface.
Returns [] if an error occurs

currentChannel(*self*, *new_channel*)

tries to set the ap channel.

Note: to confim that the channel was changed, issue currentChannel()

frequency(*self*, *new_frequency*)

not implemented yet

same as currentChannel() but uses the frequency instead

tx_bitrates(*self*, *tx_bitrates*)

not implemented yet

powerSaveMode(*self*, *new_mode*)

sets the power mode of the ap to (on or off)

fragmentationThreshold(*self*, *new_threshold*)

not implemented yet

channelBandwitdh(*self*, *new_chbw*)

not implemented yet

channelInfo(*self*)

uses MSG_GET_CHANNELINFO to get information for each channel available for the wireless interface

Return Value

a list with channel info – active_time, busy_time, channel_type, extension_channel_busy_time, frequency, in_use, noise, receive_time, transmit_time

wireless_interfaces(*self*)

get a list of all wireless interfaces

Return Value

list of interfaces

fastBSSTransition(*self*)

connect to ap requesting if it is "Fast BSS Transition" compatible

beaconInterval(*self*, *value*=`100`)

connect to AP to set beacon interval value returns nothing

getWirelessInterfaceInfo(*self*)

call ap to get information about this interface

getLinkStatitics(*self*)

not implemented yet

getACS(*self*, *num_tests*=`1`)

request that the AP computes the ACS factor for each frequency in the intf_name interface

define_msg_to_capture(*self*, *rules*, *func*)

this register in the AP rules to send all matched wireless messages to the Ethanol controller

Parameters

`func`: handler function for this messages the function has one parameter: func(received_frame)

`rules`: a list of rules - each rule identifies a type of wireless frame that should be sent to the controller

send_frame(*self*, *frame*)

calls the AP so it sends the frame

Parameters

`frame`: fully formatted (binary) frame to be sent by the AP

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

7.1.2 Properties

Name	Description
Inherited from object	
___class___	

8 Module ethanol.ethanol.station

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

8.1 Functions

add_station(*client_address*)

Create (and return) possibly several objects, one for each wireless connections identified by (client_address, interface name). This function updates a list of these objects.

client_address = (ip, port) used by the Hello message's process

get_station_by_mac_address(*mac_address*)

returns a connected station (object), provided its mac address

get_station_by_ip(*ip*)

returns a dictionary containing the connected station, provided its ip address note: that the object are indexed by the intf_name, in case the station has multiple wireless interfaces e.g. list_of_stations[ip]['wlan0']

is_sta_with_ip_connected(*ip*)

Return Value

TRUE if an STA with the ip provided as a parameter is connected

Note: this is the ip of the STA's interface that sends packets to the controller, i.e., normally it is an ethernet interface

8.2 Variables

Name	Description
list_of_stations	**Value:** `{}`

8.3 Class Station

pox.ethanol.ethanol.device.Device
└ **ethanol.ethanol.station.Station**

This module contains the Station class. Its objects represent each user connected to the VAP Each station is identified by its ip address and wireless interface name

8.3.1 Methods

__init__(*self*, *socket*, *intf_name*=`'wlan0'`)

constructor: creates an object that represents the user connection receives an ip/port pair from the hello message uses this info to connect to the station and retrieve the radio it is connected to

__del__(*self*)

destructor

vap(*self*)

the VAP the station is connected to

radio(*self*)

this station is connected to radio, if radio is None the AP is not ethanol enabled

wireless_interfaces(*self*)

returns all wireless enabled interfaces of the device

getInterferenceMap(*self*)

not implemented yet

getChannelInfo(*self*)

not implemented yet

getBeaconInfo(*self*)

not implemented yet

getNoiseInfo(*self*)

not implemented yet

getLinkMeasurement(*self*)

not implemented yet

getStatistics(*self*)

not implemented yet

getLocation(*self*)

not implemented yet

triggerTransition(*self*, *new_vap*)

uses message MSG_TRIGGER_TRANSITION to send to the station a command to change to a new ap

Parameters

`new_ap`: MAC address of the new AP

___str___(*self*)

string representation of this station

9 Module ethanol.ethanol.switch

An L2 learning switch based on L2 learning example from POX

9.1 Functions

launch(*transparent*=`False`, *hold_down*=`_flood_delay`)

Starts an L2 learning switch.

9.2 Variables

Name	Description
log	**Value:** `core.getLogger()`

9.3 Class LearningSwitch

object
└ **ethanol.ethanol.switch.LearningSwitch**

9.3.1 Methods

__init__(*self*, *connection*, *transparent*, *idle_timeout*=`10`, *hard_timeout*=`30`)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

9.3.2 Properties

Name	Description
Inherited from object	

continued on next page

Name	Description
___class___	

9.4 Class l2_learning

object ─┐
ethanol.ethanol.switch.l2_learning

Waits for OpenFlow switches to connect and makes them learning switches.

9.4.1 Methods

___init___(*self*, *transparent*)

x.___init___(...) initializes x; see help(type(x)) for signature

Overrides: object.___init___ extit(inherited documentation)

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

9.4.2 Properties

Name	Description
Inherited from object	
___class___	

10 Module ethanol.ethanol.vap

This module provides: class VAP

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

10.1 Class VAP

pox.ethanol.ethanol.device.Device ─┐

ethanol.ethanol.vap.VAP

represents the logical AP (defined by the SSID it contains) inherits DEVICE class

10.1.1 Methods

__init__(*self*, *server*, *ssid*, *radio*, *mac_address*)

constructor:

__del__(*self*)

destructor: not implemented yet

__str__(*self*)

vap string representation

register_station(*self*, *station*=`None`)

register a station in the list called by station.__init__

unregister_station(*self*, *station*)

register a station in the list called by station.__del__

stations(*self*)

return the stations (objects) currently connected to the VAP and to the controller (ethanol enabled stations)

radio(*self*)

the radio to which the radio is connected

enabled(*self*, *value*)

ssid(*self*, *value*)

change the vap's SSID

broadcastSSID(*self*, *value*)

not implemented yet

fastBSSTransitionEnabled(*self*)

not implemented yet

security(*self*)

not implemented yet

contention(*self*)

not implemented yet

cac(*self*)

not implemented yet

frameBurstEnabled(*self*)

:return if AP has frame burst feature enabled

guardInterval(*self*)

:return Guard Interval

dtimInterval(*self*)

:return DTIM interval

ctsProtection_enabled(*self*)

not implemented yet

rtsThreshold(*self*)

get RTS threshold, if 0 RTS/CTS is not used

getStationInRange(*self*)

not implemented yet

evUserConnecting(*self*, *mac_station*)

evUserAssociating(*self*, *mac_station*)

evUserAuthenticating(*self*, *mac_station*)

evUserDisassociating(*self*, *mac_station*)

evUserReassociating(*self*, *mac_station*)

evUserDisconnecting(*self*, *mac_station*)

disassociateUser(*self*, *station*)

not implemented yet

deauthenticateUser(*self*)

not implemented yet

evFastTransition(*self*)

not implemented yet

evFastReassociation(*self*)

not implemented yet

program_ProbeRequest_Interval(*self*, *Interval*=`None`)

not implemented yet

evProbeRequestReceived(*self*)

not implemented yet

evMgmtFrameReceived(*self*, *msg_type*, *msg*)

```

not implemented yet
:param msg_type indicates the type of the management frame. definition are in ieee8
        #define IEEE80211_STYPE_ASSOC_REQ   0x0000
        #define IEEE80211_STYPE_ASSOC_RESP  0x0010
        #define IEEE80211_STYPE_REASSOC_REQ 0x0020
        #define IEEE80211_STYPE_REASSOC_RESP    0x0030
        #define IEEE80211_STYPE_PROBE_REQ   0x0040
        #define IEEE80211_STYPE_PROBE_RESP  0x0050
        #define IEEE80211_STYPE_BEACON      0x0080
        #define IEEE80211_STYPE_ATIM        0x0090
        #define IEEE80211_STYPE_DISASSOC    0x00A0
        #define IEEE80211_STYPE_AUTH        0x00B0
        #define IEEE80211_STYPE_DEAUTH      0x00C0
        #define IEEE80211_STYPE_ACTION      0x00D0
:param msg message received
```

registerMgmtFrame(*self*, *msg_type*, *listener*)

unregisterMgmtFrame(*self*, *msg_type*)

not implemented yet inform the AP that it does not need to send information back to the controller about this type of message

connectNewUser(*self*, *station*, *old_ap*)

not implemented yet transfer information about a station from old_ap to this ap

connected_stations(*self*)

:return: list of stations MAC address

11 Package ethanol.events

Version: 0.3

11.1 Modules

- **events**: Events ~~~~~~
  (Section 12, p. 34)
- **tests** *(Section 13, p. 37)*
  - **tests** *(Section 14, p. 38)*

11.2 Class Events

```
Encapsulates the core to event subscription and event firing, and feels
like a "natural" part of the language.

The class Events is there mainly for 3 reasons:

    - Events (Slots) are added automatically, so there is no need to
    declare/create them separately. This is great for prototyping. (Note
    that ‘__events__‘ is optional and should primarilly help detect
    misspelled event names.)
    - To provide (and encapsulate) some level of introspection.
    - To "steel the name" and hereby remove unneeded redundancy in a call
    like:

        xxx.OnChange = event(’OnChange’)
```

11.2.1 Methods

__init__(*self*, *events*=None)

__getattr__(*self*, *name*)

__repr__(*self*)

__str__(*self*)

__len__(*self*)

___iter___(*self*)

11.3 Class EventsException

```
object ─┐
        │
exceptions.BaseException ─┐
                          │
          exceptions.Exception ─┐
                                │
                                ethanol.events.events.EventsException
```

11.3.1 Methods

Inherited from exceptions.Exception

___init___(), ___new___()

Inherited from exceptions.BaseException

___delattr___(), ___getattribute___(), ___getitem___(), ___getslice___(), ___reduce___(), ___repr___(), ___setattr___(), ___setstate___(), ___str___(), ___unicode___()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

11.3.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object ___class___	

12 Module ethanol.events.events

```
Events
~~~~~~

Implements C#-Style Events.

Derived from the original work by Zoran Isailovski:
http://code.activestate.com/recipes/410686/ - Copyright (c) 2005

:copyright: (c) 2014-2017 by Nicola Iarocci.
:license: BSD, see LICENSE for more details.
```

12.1 Variables

Name	Description
__package__	**Value:** `None`

12.2 Class EventsException

```
object ─┐
        │
exceptions.BaseException ─┐
                          │
          exceptions.Exception ─┐
                                │
                                ethanol.events.events.EventsException
```

12.2.1 Methods

Inherited from exceptions.Exception

__init__(), __new__()

Inherited from exceptions.BaseException

__delattr__(), __getattribute__(), __getitem__(), __getslice__(), __reduce__(), __repr__(), __setattr__(), __setstate__(), __str__(), __unicode__()

Inherited from object

___format___(), ___hash___(), ___reduce_ex___(), ___sizeof___(), ___subclasshook___()

12.2.2 Properties

Name	Description
Inherited from exceptions.BaseException args, message	
Inherited from object ___class___	

12.3 Class Events

```
Encapsulates the core to event subscription and event firing, and feels
like a "natural" part of the language.

The class Events is there mainly for 3 reasons:

    - Events (Slots) are added automatically, so there is no need to
    declare/create them separately. This is great for prototyping. (Note
    that ‘__events__‘ is optional and should primarilly help detect
    misspelled event names.)
    - To provide (and encapsulate) some level of introspection.
    - To "steel the name" and hereby remove unneeded redundancy in a call
    like:

        xxx.OnChange = event('OnChange')
```

12.3.1 Methods

___init___(*self*, *events*=None)

___getattr___(*self*, *name*)

___repr___(*self*)

___str___(*self*)

___len___(*self*)

__iter__(*self*)

13 Package ethanol.events.tests

13.1 Modules

- **tests** *(Section 14, p. 38)*

13.2 Variables

Name	Description
___package___	**Value:** `None`

14 Module ethanol.events.tests.tests

14.1 Variables

Name	Description
__package__	**Value:** `'ethanol.events.tests'`

14.2 Class TestBase

object
unittest.case.TestCase
ethanol.events.tests.tests.TestBase

Known Subclasses: ethanol.events.tests.tests.TestEventSlot, ethanol.events.tests.tests.TestEvents, ethanol.events.tests.tests.TestInstanceEvents

14.2.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.case.TestCase.setUp extit(inherited documentation)

callback1(*self*)

callback2(*self*)

callback3(*self*)

Inherited from unittest.case.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), addCleanup(), addTypeEqualityFunc(), assertAlmostEqual(), assertAlmostEquals(), assertDictContainsSubset(), assertDictEqual(), assertEqual(), assertEquals(), assertFalse(), assertGreater(), assertGreaterEqual(), assertIn(), assertIs(), assertIsInstance(), assertIsNone(), assertIsNot(), assertIsNotNone(), assertItemsEqual(), assertLess(), assertLessEqual(), assertListEqual(), assertMultiLineEqual(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertNotIn(), assertNotIsInstance(), assertNotRegexpMatches(), assertRaises(), assertRaisesRegexp(), assertRegexpMatches(), assertSequenceEqual(), assertSetEqual(), assertTrue(), assertTupleEqual(), assert_(), countTestCases(), debug(), defaultTestResult(), doCleanups(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUpClass(), shortDescription(), skipTest(), tearDown(), tearDownClass()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

14.2.2 Properties

Name	Description
Inherited from object	
__class__	

14.2.3 Class Variables

Name	Description
Inherited from unittest.case.TestCase	
longMessage, maxDiff	

14.3 Class TestEvents

```
object ─┐
        │
unittest.case.TestCase ─┐
                        │
ethanol.events.tests.tests.TestBase ─┐
                                     │
                                     ethanol.events.tests.tests.TestEvents
```

14.3.1 Methods

test_getattr(*self*)

test_len(*self*)

test__iter(*self*)

Inherited from ethanol.events.tests.tests.TestBase(Section 14.2)

callback1(), callback2(), callback3(), setUp()

Inherited from unittest.case.TestCase

___call___(), ___eq___(), ___hash___(), ___init___(), ___ne___(), ___repr___(), ___str___(), addCleanup(), addTypeEqualityFunc(), assertAlmostEqual(), assertAlmostEquals(), assertDictContainsSubset(), assertDictEqual(), assertEqual(), assertEquals(), assertFalse(), assertGreater(), assertGreaterEqual(), assertIn(), assertIs(), assertIsInstance(), assertIsNone(), assertIsNot(), assertIsNotNone(), assertItemsEqual(), assertLess(), assertLessEqual(), assertListEqual(), assertMultiLineEqual(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertNotIn(), assertNotIsInstance(), assertNotRegexpMatches(), assertRaises(), assertRaisesRegexp(), assertRegexpMatches(), assertSequenceEqual(), assertSetEqual(), assertTrue(), assertTupleEqual(), assert_(), countTestCases(), debug(), defaultTestResult(), doCleanups(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUpClass(), shortDescription(), skipTest(), tearDown(), tearDownClass()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___setattr___(), ___sizeof___(), ___subclasshook___()

14.3.2 Properties

Name	Description
Inherited from object	
___class___	

14.3.3 Class Variables

Name	Description
Inherited from unittest.case.TestCase	
longMessage, maxDiff	

14.4 Class TestEventSlot

```
object ─┐
        │
unittest.case.TestCase ─┐
                        │
ethanol.events.tests.tests.TestBase ─┐
                                     │
                                     ethanol.events.tests.tests.TestEventSlot
```

14.4.1 Methods

setUp(*self*)

Hook method for setting up the test fixture before exercising it.

Overrides: unittest.case.TestCase.setUp extit(inherited documentation)

test_type(*self*)

test_len(*self*)

test_repr(*self*)

test_iter(*self*)

test_getitem(*self*)

test_isub(*self*)

Inherited from ethanol.events.tests.tests.TestBase(Section 14.2)

callback1(), callback2(), callback3()

Inherited from unittest.case.TestCase

___call___(), ___eq___(), ___hash___(), ___init___(), ___ne___(), ___repr___(), ___str___(), addCleanup(), addTypeEqualityFunc(), assertAlmostEqual(), assertAlmostEquals(), assertDictContainsSubset(), assertDictEqual(), assertEqual(), assertEquals(), assertFalse(), assertGreater(), assertGreaterEqual(), assertIn(), assertIs(), assertIsInstance(), assertIsNone(), assertIsNot(), assertIsNotNone(), assertItemsEqual(), assertLess(), assertLessEqual(), assertListEqual(), assertMultiLineEqual(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertNotIn(), assertNotIsInstance(), assertNotRegexpMatches(), assertRaises(), assertRaisesRegexp(), assertRegexpMatches(), assertSequenceEqual(), assertSetEqual(), assertTrue(), assertTupleEqual(), assert_(), countTestCases(), debug(), defaultTestResult(), doCleanups(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnless-Raises(), id(), run(), setUpClass(), shortDescription(), skipTest(), tearDown(), tearDownClass()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

14.4.2 Properties

Name	Description
Inherited from object	
__class__	

14.4.3 Class Variables

Name	Description
Inherited from unittest.case.TestCase	
longMessage, maxDiff	

14.5 Class TestInstanceEvents

object
unittest.case.TestCase
ethanol.events.tests.tests.TestBase
ethanol.events.tests.tests.TestInstanceEvents

14.5.1 Methods

test_getattr(*self*)

test_instance_restriction(*self*)

Inherited from ethanol.events.tests.tests.TestBase(Section 14.2)

callback1(), callback2(), callback3(), setUp()

Inherited from unittest.case.TestCase

__call__(), __eq__(), __hash__(), __init__(), __ne__(), __repr__(), __str__(), addCleanup(), addTypeEqualityFunc(), assertAlmostEqual(), assertAlmostEquals(), assertDictContainsSubset(), assertDictEqual(), assertEqual(), assertEquals(), assertFalse(), assertGreater(), assertGreaterEqual(), assertIn(), assertIs(), assertIsInstance(), assertIsNone(), assertIsNot(), assertIsNotNone(), assertItemsEqual(), assertLess(), assertLessEqual(), assertListEqual(), assertMultiLineEqual(), assertNotAlmostEqual(), assertNotAlmostEquals(), assertNotEqual(), assertNotEquals(), assertNotIn(), assertNotIsInstance(), assertNotRegexpMatches(), assertRaises(), assertRaisesRegexp(), assertRegexpMatches(), assertSequenceEqual(), assertSetEqual(), assertTrue(), assertTupleEqual(), assert_(), countTestCases(), debug(), defaultTestResult(), doCleanups(), fail(), failIf(), failIfAlmostEqual(), failIfEqual(), failUnless(), failUnlessAlmostEqual(), failUnlessEqual(), failUnlessRaises(), id(), run(), setUpClass(), shortDescription(), skipTest(), tearDown(), tearDownClass()

Inherited from object

__delattr__(), __format__(), __getattribute__(), __new__(), __reduce__(), __reduce_ex__(), __setattr__(), __sizeof__(), __subclasshook__()

14.5.2 Properties

Name	Description
Inherited from object	
__class__	

14.5.3 Class Variables

Name	Description
Inherited from unittest.case.TestCase	
longMessage, maxDiff	

15 Package ethanol.graph_coloring

This package contains some exta components.

exact_color: contains an exact graph coloring algorithm

15.1 Modules

- **exact_color**: Graph coloring
  (Section 16, p. 45)

15.2 Variables

Name	Description
__package__	**Value:** `None`

16 Module ethanol.graph_coloring.exact_color

Graph coloring

Author: Henrique Moura

Change Log: April 04, 2017

Requires: networkx

16.1 Functions

assign_colors(*index_k*, *graph*, *colors*)

coloring(*index_k*, *graph*, *colors*)

alinegoritmo de colineoracao exata ref.: puntambekar

color_graph(*graph*)

read_graph(*clq_file*)

17 Module ethanol.server

This is a pox module. It should be called using pox.py.

Command sample:

./pox.py ethanol.server

Requires: construct (https://pypi.python.org/pypi/construct)

See Also: more info at msg_core.py

17.1 Functions

run_server(*server_address*='0.0.0.0', *server_port*=SERVER_PORT)

creates an Ethanol server at SERVER_PORT and activates it

launch()

registra a classe que trata as conexões dos Aps

17.2 Class ethanol_ap_server

```
object ─┐
        │
   ethanol.server.ethanol_ap_server
```

Waits for OpenFlow switches to connect and saves their information to match with Ethanol AP.

17.2.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__ extit(inherited documentation)

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

17.2.2 Properties

Name	**Description**
Inherited from object	
__class__	

18 Package ethanol.ssl_message

This package contains some components to implement Ethanol API. This module provides messaging capabilities to Ethanol using SSL sockets. This module is used by the ethanol classes.

See msg_common.py for the message types supported

18.1 Modules

- **enum** *(Section 19, p. 51)*
- **msg_acs**: implements the following messages:
  (Section 20, p. 52)
- **msg_ap_broadcastssid**: implements the following messages:
  (Section 21, p. 54)
- **msg_ap_ctsprotection_enabled**: implements the following messages:
  (Section 22, p. 56)
- **msg_ap_dtiminterval**: implements the following messages:
  (Section 23, p. 58)
- **msg_ap_frameburstenabled**: implements the following messages:
  (Section 24, p. 60)
- **msg_ap_guardinterval**: implements the following messages:
  (Section 25, p. 62)
- **msg_ap_in_range**: implements the following messages:
  (Section 26, p. 64)
- **msg_ap_interferencemap**: implements the following messages:
  (Section 27, p. 66)
- **msg_ap_modes**: implements the following messages:
  (Section 28, p. 67)
- **msg_ap_rtsthreshold**: implements the following messages:
  (Section 29, p. 68)
- **msg_ap_ssid**: implements: * get_ap_ssids
  (Section 30, p. 70)
- **msg_association**: implements:
  (Section 31, p. 72)
- **msg_beacon_interval**: handles the beacon interval information: gets or sets it.
  (Section 32, p. 74)
- **msg_bitrates**: implements the following messages:
  (Section 33, p. 76)
- **msg_bye**: implements the BYE message
  (Section 34, p. 78)
- **msg_changed_ap**: implements the following messages:
  (Section 35, p. 80)
- **msg_channelinfo**: implements the following messages:

(Section 36, p. 82)
- **msg_channels**: implements the following messages:
(Section 37, p. 84)
- **msg_common**: this modules contains important constants use throught out our implementation
(Section 38, p. 87)
- **msg_core**: All ssl_modules use python construct (https://pypi.python.org/pypi/construct).
(Section 39, p. 90)
- **msg_enabled**: implements the following messages:
(Section 40, p. 92)
- **msg_error**: error messagens
(Section 41, p. 94)
- **msg_frequency**: implements the following messages:
(Section 42, p. 95)
- **msg_handle_snr**: implements:
(Section 43, p. 97)
- **msg_hello**: basic hello message.
(Section 44, p. 100)
- **msg_interfaces**: implements the following messages:
(Section 45, p. 102)
- **msg_log**: defines if our modules will use pox.log facility or python log facility
(Section 46, p. 104)
- **msg_mean_sta_stats**: implements the following messages:
(Section 47, p. 105)
- **msg_memcpu**: implements the following messages:
(Section 48, p. 109)
- **msg_mtu_qlen**: implements: * set_txqueuelen * set_mtu
(Section 49, p. 111)
- **msg_ping**: implements:
(Section 50, p. 113)
- **msg_powersave**: implements the following messages:
(Section 51, p. 115)
- **msg_preamble**: implements: * get_preamble * set_preamble
(Section 52, p. 117)
- **msg_radio_wlans**: implements the following messages:
(Section 53, p. 119)
- **msg_sent_received**: implements the following messages:
(Section 54, p. 121)
- **msg_server**: this is creates the server, that deals with clients (aps and stations) messages the messages implemented are mapped in map_msg_to_procedure main entry to this module is: call run(server)
(Section 55, p. 126)
- **msg_snr_power**: implements the following messages:
(Section 56, p. 128)

- **msg_ssid**: implements the following messages:
  (Section 57, p. 131)
- **msg_sta_link_information**: implements the following messages:
  (Section 58, p. 133)
- **msg_sta_statistics**: implements the following messages:
  (Section 59, p. 135)
- **msg_station_trigger_transition**: implements the following messages:
  (Section 60, p. 137)
- **msg_statistics**: implements the following messages:
  (Section 61, p. 138)
- **msg_tos**: implements the following messages:
  (Section 62, p. 140)
- **msg_uptime**: implements the following messages:
  (Section 63, p. 142)
- **msg_wlan_info**: implements: * req_wlan_info(): MSG_WLAN_INFO
  (Section 64, p. 143)

18.2 Variables

Name	Description
___package___	**Value:** `None`

19 Module ethanol.ssl_message.enum

19.1 Functions

Enums(**sequential*, ***named*)

helper function - creates an enumeration

19.2 Variables

Name	Description
__package__	**Value:** `None`

19.3 Class Enum

helper function - creates an enumeration

> Number = Enum('a', 'b', 'c') > print Number.a 0

19.3.1 Methods

__init__(*self*, **keys*)

20 Module ethanol.ssl_message.msg_acs

implements the following messages:

* get_acs

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

20.1 Functions

get_acs(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`, *num_tests*=`1`)

request the ap to provide ACS information

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of a station to which this message should be relayed. If None don't relay message, server should process the request
`sta_port:`	socket port of the station
`num_tests:`	number of tests (greater than or equal to 1) that should be executed
`num_tests:`	int

20.2 Variables

Name	Description
msg_acs	**Value:** `Struct('msg_ap_in_range', Embed(msg_default), Embed(field...`
ACS_SCALE_FACTOR	**Value:** `1000000000000000000.0`

21 Module ethanol.ssl_message.msg_ap_broadcastssid

implements the following messages:

* get_acs

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

21.1 Functions

get_broadcastssid(*server*, *id*=`0`, *intf_name*=`None`, *ssid*=`None`)

verify is the interface is broadcasting the SSID

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*

set_broadcastssid(*server*, *id*=`0`, *intf_name*=`None`, *enable*=`False`, *ssid*=`None`)

enable or disable the broadcasting of the SSID

omitted fieldlist **Parameters**

`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*
`enable`:	set if the SSID should be broadcasted or if it is a hidden SSID
`enable`:	bool

21.2 Variables

Name	Description
msg_ap_broadcastssid	**Value:** `Struct('msg_ap_broadcastssid', Embed(msg_default), Embed(...`

22 Module ethanol.ssl_message.msg_ap_ctsprotection_enabled

implements the following messages:

* get_ctsprotection_enabled

* set_ctsprotection_enabled

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

22.1 Functions

get_ctsprotection_enabled(*server*, *id*=`0`, *intf_name*=`None`)

Verify if RTS/CTS mechanism is activated

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface. *(type=str)*

set_ctsprotection_enabled(*server*, *id*=`0`, *intf_name*=`None`, *enable*=`False`)

enable or disable RTS/CTS mechanism

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface. *(type=str)*
`enable:`	true activates RTS/CTS mechanism
`enable:`	bool

22.2 Variables

Name	Description
msg_ctsprotection_enabl-ed	**Value:** `Struct('ctsprotection_enabled', Embed(msg_default), Embed...`

23 Module ethanol.ssl_message.msg_ap_dtiminterval

implements the following messages:

* set_ap_dtiminterval

* get_ap_dtiminterval

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

23.1 Functions

get_ap_dtiminterval(*server*, *id*=`0`, *intf_name*=`None`)

get the DTIM interval set in the interface intf_name

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*

set_ap_dtiminterval(*server*, *id*=`0`, *intf_name*=`None`, *dtim_interval*=`100`)

set the DTIM interval of the interface intf_name

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`dtim_interval:`	DTIM interval *(type=int)*

Note: https://routerguide.net/dtim-interval-period-best-setting/

23.2 Variables

Name	Description
msg_ap_dtiminterval	**Value:** `Struct('msg_ap_dtiminterval', Embed(msg_default), Embed(f...`

24 Module ethanol.ssl_message.msg_ap_frameburstenabled

implements the following messages:

* get_ap_frameburstenabled

* set_ap_frameburstenabled

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

24.1 Functions

get_ap_frameburstenabled(*server*, *id*=`0`, *intf_name*=`None`)

if frame burst is enabled

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*

set_ap_frameburstenabled(*server*, *id*=`0`, *intf_name*=`None`, *enabled*=`False`)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`enabled:`	enables or disables frame burst *(type=bool)*

24.2 Variables

Name	Description
msg_ap_frameburstenabl-ed	**Value:** `Struct('msg_ap_frameburstenabled', Embed(msg_default), Em...`

25 Module ethanol.ssl_message.msg_ap_guardinterval

implements the following messages:

* get_ap_guardinterval

* set_ap_guardinterval

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

25.1 Functions

get_ap_guardinterval(*server*, *id*=`0`, *intf_name*=`None`)

get the guard interval set in the interface intf_name

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*

set_ap_guardinterval(*server*, *id*=`0`, *intf_name*=`None`, *guard_interval*=`100`)

set the guard interval of the interface intf_name

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`guard_interval:`	time used as guard interval between transmissions *(type=int)*

25.2 Variables

Name	Description
msg_ap_guardinterval	**Value:** `Struct('msg_ap_guardinterval', Embed(msg_default), Embed(...`

26 Module ethanol.ssl_message.msg_ap_in_range

implements the following messages:

* get_ap_in_range

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

26.1 Functions

get_ap_in_range(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

request the ap or the client to try to detect the aps in range, using 802.11 scanning capability

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg, num_aps, aps the received message (a Container), the number of aps in range, a list of aps (ap_in_range struct)

26.2 Variables

Name	Description
ap_in_range	**Value:** `Struct('ap_in_range', Embed(field_intf_name), Embed(field...`
msg_ap_in_range	**Value:** `Struct('msg_ap_in_range', Embed(msg_default), Embed(field...`

27 Module ethanol.ssl_message.msg_ap_interferencemap

implements the following messages:

* get_ap_interferenceMap

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

27.1 Functions

get_ap_interferenceMap(*server*, *m_id*=`0`, *intf_name*=`None`)

27.2 Variables

Name	Description
__package__	**Value:** `None`

28 Module ethanol.ssl_message.msg_ap_modes

implements the following messages:

* get_ap_supported_intf_modes

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

28.1 Functions

get_ap_supported_intf_modes(*server*, *m_id*=`0`, *intf_name*=`None`)

28.2 Variables

Name	Description
___package___	**Value:** `None`

29 Module ethanol.ssl_message.msg_ap_rtsthreshold

implements the following messages:

* get_ap_rtsthreshold

* set_ap_rtsthreshold

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

29.1 Functions

get_ap_rtsthreshold(*server*, *id*=`0`, *intf_name*=`None`)

verify is the interface is broadcasting the SSID

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*

Return Value

msg, value

set_ap_rtsthreshold(*server*, *id*=`0`, *intf_name*=`None`, *rts_threshold*=`0`)

enable or disable the broadcasting of the SSID

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*

29.2 Variables

Name	Description
msg_ap_rtsthreshold	**Value:** `Struct('msg_ap_rtsthreshold', Embed(msg_default), Embed(f...`

30 Module ethanol.ssl_message.msg_ap_ssid

implements: * get_ap_ssids

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

30.1 Functions

get_ap_ssids(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_names*=`[]`)

returns the channel and frequency of the ssid for each intf_names

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_names:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

30.2 Variables

Name	Description
ssid_info	information about the configured SSID: wiphy, ESSID, channel, frequency, mode **Value:** `Struct('ssid_info', Embed(field_intf_name), Embed(field_s...`

continued on next page

Name	Description
msg_ap_ssid	message structure **Value:** `Struct('msg_ap_ssid', Embed(msg_default), Embed(field_sta...`

31 Module ethanol.ssl_message.msg_association

implements:

* the default process function used by the controller

* process_association()

* get_association()

* register_functions() used in VAP

* set_event_association()

omitted fieldlist **Author:** Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

31.1 Functions

get_association(*server*, *id*=`0`, *association_type*=`None`, *mac_sta*=`None`, *mac_ap*=`None`)

only for tests. the controller don't use this!!!

register_functions(*mac*, *vap*)

use this function to register the VAP object process_association will call the object's methods to deal with each one of the association steps

process_association(*received_msg*, *fromaddr*)

set_event_association(*server*, *id*=`0`, *mac_sta*=`None`, *events*=`[]`, *action*=`True`)

31.2 Variables

Name	Description
field_mac_ap	handles the ap's mac address used in msg_association **Value:** `Struct('mac_ap', SLInt32('mac_ap_size'), If(lambda ctx: c...`
field_mac_sta	handles the station's mac address used in msg_association **Value:** `Struct('mac_sta', SLInt32('mac_sta_size'), If(lambda ctx:...`
msg_association	all association message types are the same, and use msg_association struct to send information **Value:** `Struct('msg_association', Embed(msg_default), Embed(field...`
registered_functions	**Value:** `{}`
EVENT_MSG_ASSOCI-ATION	**Value:** `1 << 0`
EVENT_MSG_DISASSO-CIATION	**Value:** `1 << 1`
EVENT_MSG_REASSO-CIATION	**Value:** `1 << 2`
EVENT_MSG_AUTHO-RIZATION	**Value:** `1 << 3`
EVENT_MSG_USER_D-ISCONNECTING	**Value:** `1 << 4`
EVENT_MSG_USER_C-ONNECTING	**Value:** `1 << 5`
msg_event_association	**Value:** `Struct('msg_event_association', Embed(msg_default), Embed...`

32 Module ethanol.ssl_message.msg_beacon_interval

handles the beacon interval information: gets or sets it. Implements:

* get_beacon_interval()

* set_beacon_interval()

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

32.1 Functions

get_beacon_interval(*server*, *id*=`0`, *intf_name*=`None`)

get beacon interval in miliseconds for the interface intf_name

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*

Return Value

-1 if an error occurs

set_beacon_interval(*server*, *id*=`0`, *intf_name*=`None`, *beacon_interval*=`100`)

set the beacon interval (in ms) default = 100ms different brands and models offer different allowable beacon interval ranges

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`beacon_interval:`	*(type=int)*

32.2 Variables

Name	Description
msg_beacon_interval	**Value:** `Struct('msg_beacon_interval', Embed(msg_default), Embed(f...`
ERROR	**Value:** `-1`

33 Module ethanol.ssl_message.msg_bitrates

implements the following messages:

* MSG_GET_TX_BITRATES: get_tx_bitrates

* MSG_GET_TX_BITRATE : get_tx_bitrate

* MSG_SET_TX_BITRATES: TODO

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

33.1 Functions

get_tx_bitrates(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

get the channels the interface intf_name supports, this function applies to access points

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

a dictionary, the index is the band

get_tx_bitrate(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`, *sta_mac*=`None`)

get the channels the interface intf_name supports, applies to access points

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*
`sta_mac:`	if None, scan for all stations. If specified (str with MAC address dotted format), returns only the station, if connected

33.2 Variables

Name	Description
iw_bitrates	**Value:** `Struct('iw_bitrates', LFloat32("bitrate"), ULInt8('is_sho...`
iw_bands	**Value:** `Struct('iw_bands', Embed(field_intf_name), ULInt32('band'...`
msg_tx_bitrates	**Value:** `Struct('msg_tx_bitrates', Embed(msg_default), Embed(field...`
msg_tx_bitrate	************************ MSG_TYPE.MSG_GET_TX_BITRATE ************************ **Value:** `Struct('msg_tx_bitrate', Embed(msg_default), Embed(field_...`

34 Module ethanol.ssl_message.msg_bye

implements the BYE message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

34.1 Functions

send_msg_bye(*server*, *id*=`0`, *tcp_port*=`None`)

disconnects the ethanol device from the controller

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`tcp_port:`	socket port number of the device *(type=int)*

process_bye(*received_msg*, *fromaddr*)

returns the message to the ssl server process. nothing to be done, only send back the same message

Parameters

`func_bye:`	event

bogus_bye_on_change(***kwargs*)

34.2 Variables

Name	Description
events_bye	to handle a receiving bye messages, just add your function to events_bye your function must use 'def my_funct(**kwargs)' signature for compatibility **Value:** `Events()`
msg_bye	**Value:** `Struct('msg_bye', Embed(msg_default), SLInt32('tcp_port'),)`

35 Module ethanol.ssl_message.msg_changed_ap

implements the following messages:

* changed_ap

* process_changed_ap

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

35.1 Functions

changed_ap(*server*, *id*=`0`, *status*=`0`, *current_ap*=`None`, *intf_name*=`None`)

verify is the interface is broadcasting the SSID

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`status:`	inform the status of the operation (result from change ap operation) *(type=int)*
`current_ap:`	MAC address of the ap *(type=str)*

process_hello(*received_msg*, *fromaddr*)

for now, only logs the information

Parameters

`received_msg:`	stream of bytes to be decoded
`fromaddr:`	IP address from the device that sent this message

35.2 Variables

Name	Description
field_current_ap	**Value:** `Struct('current_ap', SLInt32('current_ap_size'), If(lambd...`
msg_changed_ap	**Value:** `Struct('msg_changed_ap', Embed(msg_default), Embed(field_...`

36 Module ethanol.ssl_message.msg_channelinfo

implements the following messages:

* MSG_GET_CHANNELINFO: get_channelinfo

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

36.1 Functions

get_channelinfo(*server*, *id*=`0`, *intf_name*=`None`, *channel*=`0`, *only_channel_in_use*=`False`)

get the channels the interface inff_name supports, this function applies to access points

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`channel:`	specify a channel to scan *(type=int)*
`only_channel_in_use:`	return only the channel in use *(type=bool)*

Return Value

msg - received message a list

36.2 Variables

Name	Description
channel_info	**Value:** `Struct('channel_info', ULInt32('frequency'), SLInt8('in_u...`
msg_channelinfo	**Value:** `Struct('msg_channelinfo', Embed(msg_default), Embed(field...`

37 Module ethanol.ssl_message.msg_channels

implements the following messages:

* get_channels

* get_currentchannel

* set_currentchannel

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

37.1 Functions

get_channels(*server*, *id*=`0`, *intf_name*=`None`)

get the channels the interface inff_name supports, applies to access points

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	names of the wireless interface *(type=list of str)*

Return Value

msg - received message

get_currentchannel(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

get the channel the interface is configured to use . You can ask the AP to relay this request to the station if (sta_ip, sta_port) is provided

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

set_currentchannel(*server*, *id*=`0`, *channel*=`None`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

set the current channel to channel

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

37.2 Variables

Name	Description
valid_channel	**Value:** `Struct('valid_channel', ULInt32('frequency'), ULInt32('ch...`
msg_channels	**Value:** `Struct('msg_channels', Embed(msg_default), Embed(field_in...`
msg_currentchannel	**Value:** `Struct('msg_currentchannel', Embed(msg_default), Embed(fi...`

38 Module ethanol.ssl_message.msg_common

this modules contains important constants use throught out our implementation

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

38.1 Functions

tri_boolean(*v*, *d*)

hexadecimal(*s*)

converts a string of bytes to a string of hexa

connect_ssl_socket(*server*)

creates a ssl socket to server

Parameters

`server`: is a tuple (ip, port)

is_error_msg(*received_msg*)

get_error_msg(*received_msg*)

send_and_receive_msg(*server*, *msg_struct*, *builder*, *parser*, *only_send*=`False`)

generic function to send and receive message

Parameters

`server:`	(serverIp, serverPort)
`msg_struct:`	Container with message fields
`builder:`	Struct.build
`parser:`	Struc.parse this Struct class must be able to interpret Cointainer fields

Return Value

error : false if something goes wrong msg : a Container with the message

len_of_string(*v*)

return_from_dict(*d*, *v*, *error*)

38.2 Variables

Name	Description
VERSION	ethanol version **Value:** `"1.0.3"`
MSG_TYPE	contains all constants used as message type **Value:** `Enum('MSG_HELLO_TYPE', 'MSG_BYE_TYPE', 'MSG_ERR_TYPE', 'M...`
SERVER_ADDR	this is the default address our server is going to bind for tests, connect only to the loopback interface if you want to connect to all available interfaces, use "0.0.0.0" **Value:** `"localhost"`
SERVER_PORT	this is the default port used in the AP the port in the station is SERVER_PORT+1 (by default) **Value:** `22222`
BUFFER_SIZE	size of the buffer used by the python socket **Value:** `65536`

continued on next page

Name	Description
MSG_ERROR_TYPE	constantes usadas para definição de erro de mensagens usadas no campo error_type in msg_error.py **Value:** `Enum('ERROR_UNKNOWN', 'ERROR_VERSION_MISMATCH', 'ERROR_PR...`
DEFAULT_WIFI_INTF-NAME	**Value:** `'wlan0'`

39 Module ethanol.ssl_message.msg_core

All ssl_modules use python construct (https://pypi.python.org/pypi/construct). To install this module:

wget -c https://pypi.python.org/packages/source/c/construct/construct-2.5.2.tar.gz tar zxvf construct-2.5.2.tar.gz cd construct-2.5.2 sudo ./setup.py install

See Also: documentation at http://construct.readthedocs.io/en/latest/

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

39.1 Functions

toHex(*s*)

Parameters

`s`: is a number stored in an string

Return Value

a string, each byte of s is coded as a two char hex string

int32_to_bytes(*i*, *endian*=`'l'`)

helper function to BooleanFlag() returns boolean value coded as string of 4 bytes default is little endian

BooleanFlag(*name*, *truth_value*=`1`, *false_value*=`0`, *default*=`False`)

Defines a Construct boolean type. The flag is coded as a 32 bit value

decode_default_fields(*received_msg*)

handles the default header of all ethanol's messages

Parameters

`received_msg`: byte stream to be decoded (parsed) using construct message struct

39.2 Variables

Name	Description
msg_default	default message structure to be embedded in the first part of every message **Value:** `Struct('msg_default')`
field_intf_name	handles an interface name field (a C char * field) **Value:** `Struct('intf_name')`
field_mac_addr	handles a mac address field (a C char * field) **Value:** `Struct('mac_addr')`
field_ssid	handles a ssid field (a C char * field) **Value:** `Struct('ssid')`
field_station	handles a station IP address (a C char * field), and its port (a C int field) **Value:** `Struct('station_connection')`
___package___	**Value:** `'ethanol.ssl_message'`

40 Module ethanol.ssl_message.msg_enabled

implements the following messages:

* is_802_11e_enabled

* is_fastbsstransition_compatible

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

40.1 Functions

is_802_11e_enabled(*server*, *id*=`0`, *intf_name*=`DEFAULT_WIFI_INTFNAME`, *sta_ip*=`None`, *sta_port*=`0`)

verifies if 802.11e is supported and is enabled

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

is_fastbsstransition_compatible(*server*, *id*=`0`, *intf_name*=`DEFAULT_WIFI_INTFNAME`, *sta_ip*=`None`, *sta_port*=`0`)

checks if the interface supports fast BSS transition feature

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

40.2 Variables

Name	Description
msg_enabled	**Value:** `Struct('msg_enabled', Embed(msg_default), Embed(field_int...`

41 Module ethanol.ssl_message.msg_error

error messagens

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

41.1 Functions

return_error_msg_struct(*m_id*, *error_type*=`MSG_ERROR_TYPE.ERROR_UNKNOWN`)

return error message as an array of bytes

Parameters

`id`: message id

Return Value

msg - received message

process_msg_not_implemented(*received_msg*, *fromaddr*)

generates an error message for the case where the process procedure is not implemented in Python returns an error

(not implemented)

41.2 Variables

Name	Description
msg_error	**Value:** `Struct('msg_error', Embed(msg_default), SLInt32('error_ty...`

42 Module ethanol.ssl_message.msg_frequency

implements the following messages:

* get_frequency

* set_frequency

no process is implemented: the controller is not supposed to answer these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

42.1 Functions

get_frequency(*server*, *id*=0, *intf_name*=None, *sta_ip*=None, *sta_port*=0)

```
the interface is configured to use the frequency returned by this function
  can ask the AP to relay this request to the station if (sta_ip, sta_port) is provi

@param server: tuple (ip, port_num)
@param id: message id
@param intf_name: name of the wireless interface
@type intf_name: str
@param sta_ip: ip address of the station that this message should be relayed to, if
@type sta_ip: str
@param sta_port: socket port number of the station
@type sta_port: int

@return: msg - received message
```

set_currentchannel(*server*, *id*=`0`, *frequency*=`None`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

set the current frequency to value provided by the parameter "frequency"

Parameters

`frequency`: new channel based on frequency

(type=int)

`server`: tuple (ip, port_num)

`id`: message id

`intf_name`: name of the wireless interface

(type=str)

`sta_ip`: ip address of the station that this message should be relayed to, if sta_ip is different from None

(type=str)

`sta_port`: socket port number of the station

(type=int)

Return Value

msg - received message

42.2 Variables

Name	Description
msg_frequency	**Value:** `Struct('msg_frequency', Embed(msg_default), Embed(field_s...`

43 Module ethanol.ssl_message.msg_handle_snr

implements:

* snr_threshold_interval_reached and process_snr_threshold

* set_snr_threshold_interval

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

43.1 Functions

snr_threshold_reached(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *sta_mac*=`None`, *intf_name*=`None`, *mac_ap*=`None`, *snr*=`None`)

send information to controller. this implementation will 'never' be used in its python form

Parameters

`server`: tuple (ip, port_num)

`id`: message id

process_snr_threshold(*received_msg*, *fromaddr*)

bogus_snr_threshold_reached_on_change(***kwargs*)

snr_threshold_interval_reached(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`, *interval*=`10`)

set the time between SNR scans in the station.

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`interval:`	interval in miliseconds *(type=int)*

set_snr_threshold(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`, *threshold*=`10`)

set the SNR threshold in dBm. Send message to a station.

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`threshold:`	SNR threshold in dBm

43.2 Variables

Name	Description
events_snr_threshold_reached	to handle a receiving snr_threshold_reached message, just add your function to events_snr_threshold_reached your function must use 'def my_funct(**kwargs)' signature for compatibility **Value:** `Events()`
field_mac_ap	handles a mac address field for the new ap (a C char * field) **Value:** `Struct('mac_ap', SLInt32('mac_ap_size'), If(lambda ctx: c...`
msg_snr_threshold_reached	message structure MSG_SET_SNR_THRESHOLD_REACHED **Value:** `Struct('msg_snr_threshold_reached', Embed(msg_default), E...`

continued on next page

Name	Description
msg_snr_interval	message structure MSG_SET_SNR_INTERVAL **Value:** `Struct('msg_snr_interval', Embed(msg_default), Embed(fiel...`
msg_snr_threshold	message structure MSG_SET_SNR_THRESHOLD **Value:** `Struct('msg_snr_threshold', Embed(msg_default), Embed(fie...`

44 Module ethanol.ssl_message.msg_hello

basic hello message. Hello carries information about the ap or station to the controller

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

44.1 Functions

send_msg_hello(*server*, *m_id*=`0`)

Parameters

`server`: tuple (ip, port_num)

`m_id`: message id

Return Value

msg - received message

process_hello(*received_msg*, *fromaddr*)

returns the message to the ssl server process

Parameters

`received_msg`:

`fromaddr`: ip address of the device that sent this message

bogus_hello_on_change(***kwargs*)

44.2 Variables

Name	Description
events_hello	to handle a receiving hello message, just add your function to events_hello your function must use 'def my_funct(**kwargs)' signature for compatibility **Value:** `Events()`
msg_hello	**Value:** `Struct('msg_hello', Embed(msg_default), SLInt32('device_t...`

45 Module ethanol.ssl_message.msg_interfaces

implements the following messages:

* get_one_intf

* get_interfaces

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

45.1 Functions

get_one_intf(*server*, *m_id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

MSG_GET_ONE_INTF: eturns info of interface "intf_name"

Parameters

`server:`	tuple (ip, port_num)
`m_id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

get_interfaces(*server*, *m_id*=`0`, *sta_ip*=`None`, *sta_port*=`0`)

MSG_GET_ALL_INTF: returns all interfaces

Parameters

`server:`	tuple (ip, port_num)
`m_id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

45.2 Variables

Name	Description
intfs	**Value:** `Struct('intfs', SLInt64('ifindex'), Embed(field_intf_name...`
msg_intf	**Value:** `Struct('msg_intf', Embed(msg_default), Embed(field_statio...`

46 Module ethanol.ssl_message.msg_log

defines if our modules will use pox.log facility or python log facility

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

46.1 Variables

Name	Description
USING_POX	if true, then pox logs our module messages **Value:** `False`
__package__	**Value:** `'ethanol.ssl_message'`

47 Module ethanol.ssl_message.msg_mean_sta_stats

implements the following messages:

* send_msg_mean_sta_statistics

* send_msg_mean_sta_statistics_interface_add

* send_msg_mean_sta_statistics_interface_remove

* send_msg_mean_sta_statistics_alpha

* send_msg_mean_sta_statistics_time

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

47.1 Functions

send_msg_mean_sta_statistics(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`)

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`sta_ip`:	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port`:	socket port number of the station *(type=int)*

Return Value

msg - received message

send_msg_mean_sta_statistics_interface_add(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*
`intf_name:`	name of the wireless interface you want to get statistics from *(type=str)*

Return Value

msg - received message

send_msg_mean_sta_statistics_interface_remove(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*
`intf_name:`	name of the wireless interface you want to remove from pool *(type=str)*

Return Value

msg - received message

send_msg_mean_sta_statistics_alpha(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *alpha*=`0.1`)

Parameters

`server:` tuple (ip, port_num)

`id:` message id

`sta_ip:` ip address of the station that this message should be relayed to, if sta_ip is different from None

(type=str)

`sta_port:` socket port number of the station

(type=int)

`alpha:` alpha from EWMA

(type=float)

Return Value

msg - received message

send_msg_mean_sta_statistics_time(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *msec*=`100`)

Parameters

`server:` tuple (ip, port_num)

`id:` message id

`sta_ip:` ip address of the station that this message should be relayed to, if sta_ip is different from None

(type=str)

`sta_port:` socket port number of the station

(type=int)

`msec:` statistics are collected during "msec" interval

(type=int)

Return Value

msg - received message

47.2 Variables

Name	Description
mean_net_statistics	**Value:** `Struct('mean_net_statistics', LFloat64('collisions'), LFl...`

continued on next page

Name	Description
msg_mean_statistics	**Value:** `Struct('msg_mean_statistics', Embed(msg_default), Embed(f...`
msg_mean_sta_statistics-_interface	**Value:** `Struct('msg_mean_sta_statistics_interface', Embed(msg_def...`
msg_mean_sta_statistics-_alpha	**Value:** `Struct('msg_mean_sta_statistics_alpha', Embed(msg_default...`
msg_mean_sta_statistics-_time	**Value:** `Struct('msg_mean_sta_statistics_time', Embed(msg_default)...`

48 Module ethanol.ssl_message.msg_memcpu

implements the following messages:

* get_memory_usage

* get_cpu_usage

no process is implemented: the controller is not supposed to respond to these message

Note: see msg_cpu.h and msg_memory.h in hostapd/src/messaging

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

48.1 Functions

get_memory_usage(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`)

requests the memory usage (in percent) implements MSG_GET_MEMORY

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`sta_ip:`	ip address of a station that this message should be relayed to
`sta_port:`	socket port of the station

Return Value

msg, memory usage in percent

get_cpu_usage(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`)

requests the memory usage (in percent) implements MSG_GET_CPU

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`sta_ip:`	ip address of a station that this message should be relayed to
`sta_port:`	socket port of the station

Return Value

msg, cpu usage in percent

48.2 Variables

Name	Description
msg_memcpu	format the MSG_GET_CPU and MSG_GET_MEMORY data structure to be sent by ethanol protocol **Value:** `Struct('msg_memcpu', Embed(msg_default), Embed(field_stat...`

49 Module ethanol.ssl_message.msg_mtu_qlen

implements: * set_txqueuelen * set_mtu

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

49.1 Functions

set_msg_mtu_qlen(*server*, *m_type*, *m_id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`, *value*=`None`)

sets the MTU or Queue Len values

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*
`intf_name:`	name of the interface

set_mtu(*server*, *m_id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`, *mtu*=`None`)

set_txqueuelen(*server*, *m_id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`, *txqueuelen*=`None`)

49.2 Variables

Name	**Description**
msg_mtu_qlen	message structure **Value:** `Struct('msg_mtu_qlen', Embed(msg_default), Embed(field_st...`

50 Module ethanol.ssl_message.msg_ping

implements:

* process_msg_ping(): generates a pong message in response to a received ping message

* send_msg_ping(): send a ping to another device

Note: see msg_ping.h in hostapd/src/messaging

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

50.1 Functions

generate_ping_data(*p_size*=64)

verify_data(*data*, *p_size*)

check if the payload received is correct

send_msg(*server*, *msg*)

sends a message PING msg to the server

Parameters

`server`:	tuple (ip, port) used to socket connect to the client
`msg`:	message to be sent (ping or pong)

send_msg_ping(*server*, *id*=`0`, *num_tries*=`1`, *p_size*=`64`)

send a ping message to other ethanol device (mainly to the controller) and receives a pong response

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`num_tries:`	number of message retries before quitting
`p_size:`	payload size (extra size in bytes added to the message)

Return Value

all messages sent

process_msg_ping(*received_msg*, *fromaddr*)

grabs the ping message, verifies the data field and returns a pong message

50.2 Variables

Name	Description
msg_ping	ping message data structure **Value:** `Struct('msg_ping', Embed(msg_default), SLInt32('data_size...`
msg_pong	pong message data structure **Value:** `Struct('msg_pong', Embed(msg_default), LFloat32('rtt'), S...`
BYTE_INICIAL	**Value:** `48`

51 Module ethanol.ssl_message.msg_powersave

implements the following messages:

* get_powersave_mode(intf_name)

* set_powersave_mode(intf_name, powersave_mode)

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

51.1 Functions

get_powersave_mode(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

get if the powersave is set or not

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

set_powersave_mode(*server*, *id*=`0`, *powersave*=`True`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

51.2 Variables

Name	Description
msg_powersave	**Value:** `Struct('msg_powersave', Embed(msg_default), Embed(field_i...`

52 Module ethanol.ssl_message.msg_preamble

implements: * get_preamble * set_preamble

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

52.1 Functions

get_preamble(*server*, *id*=`0`, *intf_name*=`None`)

gets if the configured preamble is long or short

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*

Return Value

msg - received message

set_preamble(*server*, *id*=`0`, *intf_name*=`None`, *preamble*=`0`)

```
set the preamble used in some interface
  0 = preamble LONG | 1 = preamble SHORT
@param server: tuple (ip, port_num)
@param id: message id
@param intf_name: name of the wireless interface
@type intf_name: str
@param preamble:
@type sta_ip: bool

@return: msg - received message
```

52.2 Variables

Name	Description
msg_preamble	**Value:** `Struct('msg_preamble', Embed(msg_default), Embed(field_in...`

53 Module ethanol.ssl_message.msg_radio_wlans

implements the following messages:

* get_radio_wlans() : MSG_GET_RADIO_WLANS

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

53.1 Functions

get_radio_wlans(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests the radio wlans, if intf_name is not None, only this interface is considered, otherwise returns all wireless interfaces

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	name of the wireless interface *(type=str)*
`sta_ip`:	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port`:	socket port number of the station *(type=int)*

Return Value

msg - received message

53.2 Variables

Name	Description
list_of_radio_wlans	message structure **Value:** `Struct('list_of_radio_wlans', Embed(field_intf_name), Emb...`
msg_radio_wlans	**Value:** `Struct('msg_radio_wlans', Embed(msg_default), Embed(field...`

54 Module ethanol.ssl_message.msg_sent_received

implements the following messages:

* send_msg_get_bytesreceived

* send_msg_get_bytessent

* send_msg_get_byteslost

* send_msg_get_packetsreceived

* send_msg_get_packetssent

* send_msg_get_packetslost

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

54.1 Functions

send_msg_sent_received(*server*, *id*=`0`, *type*=`None`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

INTERNAL FUNCTION: don't call this function

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message value (bytes or packets received or sent or lost)

send_msg_get_bytesreceived(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests number of bytes received. this number is always incremented since the interface activation

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

send_msg_get_bytessent(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests number of bytes sent by the interface. this number is always incremented since the interface activation

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

send_msg_get_packetsreceived(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests number of packets received by the interface. this number is always incremented since the interface activation

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

send_msg_get_packetssent(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests number of packets sent by the interface. this number is always incremented since the interface activation

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

send_msg_get_packetslost(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

requests number of packets lost by the interface. this number is always incremented since the interface activation

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

54.2 Variables

Name	Description
msg_sent_received	message structure common to all supported_messages messages **Value:** `Struct('msg_sent_received', Embed(msg_default), Embed(fie...`
supported_messages	this module deals with multiple message types. these types are listed in supported_messages **Value:** `[MSG_TYPE.MSG_GET_BYTESRECEIVED, MSG_TYPE.MSG_GET_BYTESSE...`

55 Module ethanol.ssl_message.msg_server

this is creates the server, that deals with clients (aps and stations) messages the messages implemented are mapped in map_msg_to_procedure main entry to this module is: call run(server)

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

55.1 Functions

deal_with_client(*connstream*, *fromaddr*)

this function is called as a Thread to manage each connection

Parameters

`connstream`:

`fromaddr`:

run(*server*)

to use this module only call this method, providing a tuple with (server ip address, server port)

Parameters

`server`: (ip, port) tuple

55.2 Variables

Name	Description
map_msg_to_procedure	all message types supported **Value:** `{MSG_TYPE.MSG_ASSOCIATION: process_association, MSG_TYPE....`

continued on next page

Name	Description
DEFAULT_CERT_PAT-H	path to the ssl certificate used in the secure socket connections **Value:** `os.path.dirname(os.path.abspath(__file__))`
SSL_CERTIFICATE	path and default name of the ssl certificate **Value:** `DEFAULT_CERT_PATH+ '/mycert.pem'`

56 Module ethanol.ssl_message.msg_snr_power

implements the following messages:

* get_snr: MSG_GET_SNR

* get_txpower: MSG_GET_TXPOWER

* set_txpower: MSG_SET_TXPOWER

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

56.1 Functions

get_snr_power(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`, *m_type*=`None`)

INTERVAL FUNCTION: DON'T CALL THIS METHOD.

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

get_snr(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

obtain SNR

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

get_txpower(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

obtain txpower

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

set_txpower(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`, *txpower*=`None`)

set the txpower for the wireless interfacce

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	name of the wireless interface *(type=str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

56.2 Variables

Name	Description
msg_snr_power	**Value:** `Struct('msg_snr_power', Embed(msg_default), Embed(field_i...`

57 Module ethanol.ssl_message.msg_ssid

implements the following messages:

* get_ssid

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

57.1 Functions

get_ssid(*server*, *id*=`0`, *intf_name*=`[]`, *sta_ip*=`None`, *sta_port*=`0`)

returns the value None equals an error has occured (or no interface found)

Parameters

`server`: tuple (ip, port_num)

`id`: message id

`intf_name`: names of the wireless interface

(type=list of str)

`sta_ip`: ip address of the station that this message should be relayed to, if sta_ip is different from None

(type=str)

`sta_port`: socket port number of the station

(type=int)

Return Value

msg - received message

57.2 Variables

Name	Description
ssid_info	**Value:** `Struct('ssid_info', Embed(field_intf_name), Embed(field_s...`
msg_ssid	**Value:** `Struct('msg_ssid', Embed(msg_default), Embed(field_statio...`

58 Module ethanol.ssl_message.msg_sta_link_information

implements the following messages:

* get_sta_link_info

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

58.1 Functions

get_sta_link_info(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *intf_name*=`None`)

returns three values: mac_addr, ssid, frequency None equals an error has occured (or no interface found)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

To Do: Nao eh necessario retornar intf_name

58.2 Variables

Name	Description
msg_sta_link_info	**Value:** `Struct('msg_sta_link_info', Embed(msg_default), Embed(fie...`

59 Module ethanol.ssl_message.msg_sta_statistics

implements the following messages:

* get_ssid

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

59.1 Functions

get_sta_statistics(*server*, *id*=0, *intf_name*=None, *sta_ip*=None, *sta_port*=0)

returns the value None equals an error has occured (or no interface found)

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id
`intf_name`:	names of the wireless interface *(type=list of str)*
`sta_ip`:	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port`:	socket port number of the station *(type=int)*

Return Value

msg - received message

59.2 Variables

Name	Description
field_time_stamp	**Value:** `Struct('time_stamp', SLInt32('time_stamp_size'), If(lambd...`
stats_field	**Value:** `Struct('stats', Embed(field_mac_addr), Embed(field_intf_n...`
msg_sta_statistics	**Value:** `Struct('msg_sta_statistics', Embed(msg_default), Embed(fi...`

60 Module ethanol.ssl_message.msg_station_trigger_transition

implements the following messages:

* station_trigger_transition

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

60.1 Functions

station_trigger_transition(*server*, *id*=`0`, *sta_ip*=`None`, *sta_port*=`0`, *sta_mac*=`None`, *intf_name*=`None`, *mac_new_ap*=`None`)

sendo command to station to change to a new ap

Parameters

`server`:	tuple (ip, port_num)
`id`:	message id

60.2 Variables

Name	Description
field_mac_new_ap	handles a mac address field for the new ap (a C char * field) **Value:** `Struct('mac_new_ap', SLInt32('mac_new_ap_size'), If(lambd...`
msg_station_trigger_tra-nsition	message structure common to all supported_messages messages **Value:** `Struct('msg_station_trigger_transition', Embed(msg_defaul...`

61 Module ethanol.ssl_message.msg_statistics

implements the following messages:

* send_msg_get_statistics

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

61.1 Functions

send_msg_get_statistics(*server*, *id*=`0`, *intf_name*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

INTERNAL FUNCTION

returns the statistics using a dict() with 9 fields

Parameters

`server`: tuple (ip, port_num)

`id`: message id

`intf_name`: names of the wireless interface

(type=list of str)

`sta_ip`: ip address of the station that this message should be relayed to, if sta_ip is different from None

(type=str)

`sta_port`: socket port number of the station

(type=int)

Return Value

msg - received message

61.2 Variables

Name	Description
field_time_stamp	**Value:** `Struct('time_stamp', SLInt32('time_stamp_size'), If(lambd...`
msg_statistics	message structure common to all supported statistics messages **Value:** `Struct('msg_statistics', Embed(msg_default), Embed(field_...`

62 Module ethanol.ssl_message.msg_tos

implements the following messages:

* msg_tos_cleanall

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

62.1 Functions

tos_cleanall(*server*, *id*=`0`)

msg_tos_cleanall uptime

Parameters

`server`: tuple (ip, port_num)

`id`: message id

Return Value

nothing

tos_add(*server*, *msg_id*=`0`, *intf_name*=`None`, *proto*=`None`, *sip*=`None`, *sport*=`None`, *dip*=`None`, *dport*=`None`, *wmm_class*=`0`)

add TOS rule

Parameters

`server`: tuple (ip, port_num)

`msg_id`: message id

Return Value

nothing

tos_replace(*server*, *msg_id*=`0`, *rule_id*=`-1`, *intf_name*=`None`, *proto*=`None`, *sip*=`None`, *sport*=`None`, *dip*=`None`, *dport*=`None`, *wmm_class*=`0`)

msg_tos_cleanall uptime

Parameters

`server`: tuple (ip, port_num)

`id`: message id

Return Value

nothing

62.2 Variables

Name	Description
msg_tos_cleanall	message to clear mange rules **Value:** `Struct('msg_tos_cleanall', Embed(msg_default),)`
msg_tos	message to add or replace mange rules **Value:** `Struct('msg_tos', Embed(msg_default), SLInt32('rule_id'),...`

63 Module ethanol.ssl_message.msg_uptime

implements the following messages:

* get_uptime

no process is implemented: the controller is not supposed to respond to these message

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

63.1 Functions

get_uptime(*server*, *id*=0)

get uptime

Parameters

`server`: tuple (ip, port_num)

`id`: message id

Return Value

msg - received message value (bytes or packets received or sent or lost)

63.2 Variables

Name	Description
msg_uptime	message structure common to all supported_messages messages **Value:** `Struct('msg_uptime', Embed(msg_default), LFloat64('uptime...`

64 Module ethanol.ssl_message.msg_wlan_info

implements: * req_wlan_info(): MSG_WLAN_INFO

Author: Henrique Duarte Moura

Organization: WINET/DCC/UFMG

Copyright: h3dema (c) 2017

Contact: henriquemoura@hotmail.com

Since: July 2015

Status: in development

Requires: construct 2.5.2

64.1 Functions

req_wlan_info(*server*, *id*=`0`, *intf_name_list*=`None`, *sta_ip*=`None`, *sta_port*=`0`)

Parameters

`server:`	tuple (ip, port_num)
`id:`	message id
`intf_name_list:`	names of the wireless interface *(type=list of str)*
`sta_ip:`	ip address of the station that this message should be relayed to, if sta_ip is different from None *(type=str)*
`sta_port:`	socket port number of the station *(type=int)*

Return Value

msg - received message

64.2 Variables

Name	Description
wlan_entry	information about a wifi interface **Value:** `Struct('wlan_entry', SLInt32('ifindex'), Embed(field_intf...`

continued on next page

Name	Description
msg_wlan_info	**Value:** `Struct('msg_wlan_info', Embed(msg_default), Embed(field_s...`

65 Package ethanol.tos

```
This package contains some components to implement Ethanol API.
ethanol should run as a pox module

sample command call:
  python ./pox.py forwarding.l2_learning ethanol.server

ethanol.server is the ~/ethanol/python/server.py file

you must create a symbolic link inside pox subtree, like:
cd ~/ethanol/pox/pox
ln ~/ethanol/python ethanol
```

65.1 Modules

- **usecase_tos**: This is a module that runs inside POX.
  (Section 66, p. 146)

65.2 Variables

Name	Description
__package__	**Value:** `None`

66 Module ethanol.tos.usecase_tos

```
This is a module that runs inside POX.
It checks if ethanol is running,
if ethanol is running, this module can also run

 To run this module use:
 cd pox
 python pox.py log --format="[%(asctime)s] %(message)s" --datefmt="%Y-%m-%dT%H:%M:%S" et
```

66.1 Functions

set_app_paths(*paths*=[''])

set python's system path so we can call our functions :param list paths contains the relative path for our functions

is_ethanol_loaded(*module_name*='pox.ethanol.server')

```
verifies if ethanol module is loaded

Keyword Arguments:
  module_name {str} -- name of the ethanol module (default: {'pox.ethanol.server'})

Returns:
  [bool] -- True if the module is loaded
```

videocapture_traffic(*sta*, *access_class*=AC_BE, *rule_id*=1)

default_setup(*sta*, *access_class*=AC_BE)

call ethanol, to set new value

launch(*ap_ip*='150.164.10.90', *sta_ip*='150.164.10.22', *sta_intf_name*='wlan0', *hostname*='0.0.0.0', *host_port*=50000)

sta_ip contains arpias' wlan0 ip socket_address (video capture)

66.2 Variables

Name	Description
vap	**Value:** `None`
log	this is the path of this python file **Value:** `core.getLogger()`
AC_BK	**Value:** `1`
AC_BE	**Value:** `3`
AC_VI	**Value:** `5`
AC_VO	**Value:** `7`
rule1	**Value:** `{'rule_id': 1, 'intf_name': 'wlan0', 'proto': 'tcp', 'sip...`
rule2	**Value:** `{'rule_id': 2, 'intf_name': 'wlan0', 'proto': 'udp', 'sip...`

67 Script script-produce_doc_sh

Index