Eclipse fog05 Documentation

Release 0.0.1

Eclipse Foundation

Nov 13, 2019

Contents

CHAPTER 1

Installation

In order to run install Eclipse fog05 SDK you need to install `zenoh-c`, `zenoh-python` and `yaks-python` you can get those from GitHub:

```
git clone github.com/atolab/zenoh-c
cd zenoh-c
git checkout 58bad2cf1616f405fe401b22a713b95a6fef786c
make
sudo make install
cd ..
git clone github.com/atolab/zenoh-python
cd zenoh-python
git checkout 1ced877917816acea13e58c151e02cf950ad8009
sudo python3 setup.py install
cd ..
git clone github.com/atolab/yaks-python
cd yaks-python
git checkout 50c9fc7d022636433709340f220e7b58cd74cefc
sudo make install
```

Once you have those dependecies installed you can install the API:

```
pip3 install pyangbind pyang
git clone github.com/eclipse-fog05/sdk-python
cd sdk-python
make
sudo make install
```

CHAPTER 2

API

Eclipse fog05 SDK provides the following API:

2.1 Plugin API

Eclipse fog05 Plugin API

2.1.1 Plugin

class fog05_sdk.interfaces.Plugin.**Plugin**(*version*, *plugin_uuid=None*)

Class: Plugin

This class is an interface for plugins

Methods

get_agent(self)	Retrives the agent from YAKS
get_local_mgmt_address(self)	Gets local management IP address
get_nm_plugin(self)	Retrives the network manager plugin from YAKS
get_os_plugin(self)	Retrives the operating system plugin from YAKS
get_version(self)	Gets plugin version

get_agent(*self*)

Retrives the agent from YAKS

Returns

Agent

get_local_mgmt_address(*self*)

Gets local management IP address

Returns

string

get_nm_plugin(*self*)
Retrives the network manager plugin from YAKS

Returns

NM

get_os_plugin(*self*)
Retrives the operating system plugin from YAKS

Returns

OS

get_version(*self*)
Gets plugin version

Returns

string

2.1.2 OS

class fog05_sdk.interfaces.Plugin.Plugin.**OS**(*uuid*, *connector*, *node*)
Class: OS

This class encapsulates the comunication with an OS plugin using YAKS Evals

Methods

call_os_plugin_function(self, fname, fparameters)	Calls an Eval registered within the OS Plugin
check_if_pid_exists(self, pid)	Checks if a given PID exists
checksum(self, file_path)	Calculates the SHA256 checksum for the given file
create_dir(self, dir_path)	Creates the given new directory
create_file(self, file_path)	Creates the given new empty file
dir_exists(self, dir_path)	Checks if the given directory exists
download_file(self, url, file_path)	Downloads the given file in the given path
execute_command(self, command[, blocking, ...])	Executes a command on underlying os,
file_exists(self, file_path)	Checks if the given file exists
get_intf_type(self, name)	Gets the inteface type for the given interface
get_network_informations(self)	Gets information about node network interfaces
local_mgmt_address(self)	Gets node management IP address
read_file(self, file_path[, root])	Read the content from a file in the local disk, maybe can convert from windows dir separator to unix dir separator return the file content throw an exception if file not exits
remove_dir(self, dir_path)	Removes the given directory
remove_file(self, file_path)	Removes the given file
send_sig_int(self, pid)	Sends a SigKill (Ctrl+C) to the given PID

Continued on next page

Table 2 – continued from previous page

send_sig_kill(self, pid)	Sends a SigKill (kill the process) to the given pid throw an exception if pid not exits
set_interface_available(self, intf_name)	Sets a given network device as available
set_interface_unaviable(self, intf_name)	Sets a given network device as unavailable
store_file(self, content, file_path, filename)	Stores a file in local disk

call_os_plugin_function (*self*, *fname*, *fparameters*)
Calls an Eval registered within the OS Plugin

Parameters

fname [string] function name

fparameters [dictionary] parameters for the function

returns

——

whatever the fname function returns or raises a ValueError

check_if_pid_exists (*self*, *pid*)
Checks if a given PID exists

Parameters

pid [int] PID to be verified

Returns

bool

checksum (*self*, *file_path*)
Calculates the SHA256 checksum for the given file

Parameters

file_path [string] path to file

Returns

string

create_dir (*self*, *dir_path*)
Creates the given new directory

Parameters

dir_path [string] path to the directory

Returns

bool

create_file (*self*, *file_path*)
Creates the given new empty file

Parameters

file_path [string] path to the file

Returns

bool

dir_exists(*self*, *dir_path*)
Checks if the given directory exists

Parameters

dir_path [string] path to the directory

Returns

bool

download_file(*self*, *url*, *file_path*)
Downloads the given file in the given path

Parameters

url [string] url for the source file

file_path [string] path to destination file

returns

——

bool

execute_command(*self*, *command*, *blocking=False*, *external=False*)
Executes a command on underlying os,

command [string] command to be executed

blocking [bool] true if the call has to block until the end of the command

external [bool] true if the command has to be executed in an external os shell

string

file_exists(*self*, *file_path*)
Checks if the given file exists

Parameters

file_path [string] path to the file

Returns

bool

get_intf_type(*self*, *name*)
Gets the inteface type for the given interface

Parameters

name [string] name of the interface

Returns

string

get_network_informations(*self*)
Gets information about node network interfaces

Returns

list of dictionaties

{ 'intf_name':string, 'intf_mac_address':string, 'intf_speed': int, 'type':string, 'available':bool, 'default_gw':bool, 'intf_configuration': {

'ipv4_address':string, 'ipv4_netmask':string, 'ipv4_gateway':string.
'ipv6_address':string, 'ipv6_netmask':string, 'ipv6_gateway':string.
'bus_address':string

}

}

local_mgmt_address (*self*)
Gets node management IP address

Returns

string

read_file (*self*, *file_path*, *root=False*)
Read the content from a file in the local disk, maybe can convert from windows dir separator to unix dir separator return the file content throw an exception if file not exits

Parameters

file_path [string] path to file

root [bool] if true it will use sudo cat to read the file

Returns

bytes

remove_dir (*self*, *dir_path*)
Removes the given directory

Parameters

dir_path [string] path to the directory

Returns

bool

remove_file (*self*, *file_path*)
Removes the given file

Parameters

file_path [string] path to the directory

Returns

bool

send_sig_int (*self*, *pid*)
Sends a SigKill (Ctrl+C) to the given PID

Parameters

pid [int] pid to be signaled

Returns

bool

send_sig_kill (*self*, *pid*)
Sends a SigKill (kill the process) to the given pid throw an exception if pid not exits

Parameters

pid [int] pid to be signaled

Returns

bool

set_interface_available(*self*, *intf_name*)
Sets a given network device as available

Returns

bool

set_interface_unaviable(*self*, *intf_name*)
Sets a given network device as unavailable

Returns

bool

store_file(*self*, *content*, *file_path*, *filename*)
Stores a file in local disk

Parameters

content [string] file content

file_path [string] path where the content will stored

filename [string] name of the file

Returns

bool

2.1.3 NM

class fog05_sdk.interfaces.Plugin.Plugin.**NM**(*uuid*, *connector*, *node*)
Class: NM

This class encapsulates the comunication with an NM plugin using YAKS Evals

Methods

add_port_to_router(self, router_id, port_type)	Adds a port to the given virtual router
assign_floating_ip(self, ip_id, cp_id)	Assigns the given floating IP to the given conncetion point
call_nw_plugin_function(self, fname, fparameters)	Calls an Eval registered within the NM Plugin
connect_cp_to_vnetwork(self, cp_id, vnet_id)	Connects the given connection point to the given network
connect_interface_to_connection_point(self, ...)	Connects the given interace to the given connection point
create_bridges_if_not_exist(self, ...)	Creates the bridges missing after checking the manifest file
create_floating_ip(self)	Creates a floating IP
create_virtual_bridge(self, name, uuid)	Creates a virtual bridge
create_virtual_interface(self, intf_id, ...)	Creates a virtual network interface

Continued on next page

Table 3 – continued from previous page

delete_floating_ip(self, ip_id)	Deletes the given floating IP
delete_port(self, cp_id)	Deletes the given connection point
delete_virtual_bridge(self, br_uuid)	Deletes a virtual bride
delete_virtual_interface(self, intf_id)	Deletes the given virtual interface
disconnect_cp(self, cp_id)	Disconnects the given connection point
disconnect_interface(self, intf_id)	Disconnects the given interface
get_address(self, mac_address)	Gets the IP address associated to the interface with the given MAC address
get_overlay_face(self)	Gets the configured interface for overlay networks
get_vlan_face(self)	Gets the configured interface for VLAN networks
remove_floating_ip(self, ip_id, cp_id)	Retains the given floating IP from the given connection point
remove_port_from_router(self, router_id, vnet_id)	Removes a port from the given router

add_port_to_router(*self*, *router_id*, *port_type*, *vnet_id=None*, *ip_address=None*)
Adds a port to the given virtual router

Parameters

router_id [string] UUID of the virtual router

port_type [string] kind of the port to be added (INTERNAL, EXTERNAL)

vnet_id [string] eventual network to be connected

ip_address [string] eventual address for the new router port

Returns

dictionary

assign_floating_ip(*self*, *ip_id*, *cp_id*)
Assigns the given floating IP to the given conncetion point

Parameters

ip_id [string] UUID of the floating IP

cp_id [string] UUID of the connection point

Returns

dictionary

call_nw_plugin_function(*self*, *fname*, *fparameters*)
Calls an Eval registered within the NM Plugin

Parameters

fname [string] function name

fparameters [dictionary] parameters for the function

returns

——

whatever the fname function returns or raises a ValueError

connect_cp_to_vnetwork(*self*, *cp_id*, *vnet_id*)
Connects the given connection point to the given network

Parameters

cp_id [string] UUID of the connection point

vnet_id [string] UUID of the virtual network

Returns

dictionary {'int':string, 'ext':string}

connect_interface_to_connection_point(*self*, *intf_id*, *cp_id*)
Connects the given interace to the given connection point

Parameters

intf_id [string] ID of the virtual interface

cp_id [string] UUID of the connection point

Returns

dictionary {'int':string, 'ext':string}

create_bridges_if_not_exist(*self*, *expected_bridges*)
Creates the bridges missing after checking the manifest file

Parameters

expected_bridges [string list] names of expected bridges

Returns

string list

create_floating_ip(*self*)
Creates a floating IP

Returns

dictionary

create_virtual_bridge(*self*, *name*, *uuid*)
Creates a virtual bridge

Parameters

name [string] name of the virtual bridge to be created

Returns

dictionary

create_virtual_interface(*self*, *intf_id*, *descriptor*)
Creates a virtual network interface

Returns

dictionary

delete_floating_ip(*self*, *ip_id*)
Deletes the given floating IP

Parameters

ip_id [string] UUID of the floating IP

Returns

dictionary

delete_port (*self*, *cp_id*)
Deletes the given connection point

Parameters

cp_id [string] UUID of the connection point

Returns

dictionary

delete_virtual_bridge (*self*, *br_uuid*)
Deletes a virtual bride

Parameters

br_uuid [string] bridge UUID

Returns

dictionary

delete_virtual_interface (*self*, *intf_id*)
Deletes the given virtual interface

intf_id : string

Returns

dictionary

disconnect_cp (*self*, *cp_id*)
Disconnects the given connection point

Parameters

cp_id [string] UUID of connection point

Returns

dictionary {'int':string, 'ext':string}

disconnect_interface (*self*, *intf_id*)
Disconnects the given interface

Parameters

intf_id [string] ID of the virtual interace

Returns

dictionary {'int':string, 'ext':string}

get_address (*self*, *mac_address*)
Gets the IP address associated to the interface with the given MAC address

Parameters

mac_address [string] the MAC address of the interface

Returns

string

get_overlay_face (*self*)
Gets the configured interface for overlay networks

Returns

string

get_vlan_face (*self*)
Gets the configured interface for VLAN networks

Returns

string

remove_floating_ip (*self*, *ip_id*, *cp_id*)
Retains the given floating IP from the given connection point

Parameters

ip_id [string] UUID of the floating IP

cp_id [string] UUID of the connection point

Returns

dictionary

remove_port_from_router (*self*, *router_id*, *vnet_id*)
Removes a port from the given router

router_id [string] UUID of the virtual router

vnet_id [string] network to be disconnected

Returns

dictionary

2.1.4 Agent

class fog05_sdk.interfaces.Plugin.Plugin.**Agent** (*connector*, *node*)
Class: OS

This class encapsulates the comunication with Agent using YAKS Evals

Methods

call_agent_function(self, fname, fparameters)	Calls an Eval registered within the Agent
get_fdu_info(self, nodeid, fduid, instanceid)	Gets information about the given FDU instance
get_image_info(self, imageid)	Gets information about the given image
get_network_info(self, uuid)	Gets information about the given virtual network
get_node_mgmt_address(self, nodeid)	Gets management IP address for the given node
get_port_info(self, cp_uuid)	Gets information about a given connection point

call_agent_function (*self*, *fname*, *fparameters*)
Calls an Eval registered within the Agent

Parameters

fname [string] function name

fparameters [dictionary] parameters for the function

returns

——

whatever the fname function returns or raises a ValueError

get_fdu_info (*self*, *nodeid*, *fduid*, *instanceid*)
Gets information about the given FDU instance

Parameters

nodeid [string] UUID of the node

fduid [string] UUID of the FDU

instanceid [string] UUID of the instance

Returns

dictionary

get_image_info (*self*, *imageid*)
Gets information about the given image

Parameters

imageid [string] UUID of the image

Returns

dictionary

get_network_info (*self*, *uuid*)
Gets information about the given virtual network

Parameters

uuid [string] UUID of the virtual network

Returns

dictionary

get_node_mgmt_address (*self*, *nodeid*)
Gets management IP address for the given node

Parameters

nodeid [string] UUID of the node

Returns

string

get_port_info (*self*, *cp_uuid*)
Gets information about a given connection point

Parameters

cp_uuid [string] UUID of the connection point

Returns

dictionary

2.1.5 RuntimePluginFDU

CHAPTER 3

FDU Types

Eclipse fog05 SDK FDU Interfaces

3.1 FDU

3.2 InfraFDU

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

G

L

N

O

P

R

S