Eclipse fog05 Documentation

Release 0.0.1

Eclipse Foundation

Nov 13, 2019

Contents

CHAPTER 1

Installation

In order to run install Eclipse fog05 Client API you need to install `fog05_sdk` you can get those from GitHub:

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

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

```
git clone github.com/eclipse-fog05/api-python
cd api-python
sudo make install
```

CHAPTER 2

Getting Started

To kick off our tour of Eclipse fog05, we will start with the obligatory "hello world" example. This example will deploy a native compoent written in python that create a file and write "hello, world!" to the file.

Let's get started.

First, we write the python code.

```
import time

with open('/tmp/fos_helloworld', 'a') as out:
    while True:
        out.write('Hello, world! It is {}\n'.format(time.time_ns()))
        time.sleep(2)
```

Let's save it as `/tmp/fos_helloworld.py`

Next, we need to write the descriptor for this native component. Let's call it `fdu_helloworld.json`:

```
{
    "id": "helloword_fdu",
    "name": "helloworld",
    "computation_requirements": {
        "cpu_arch": "x86_64",
        "cpu_min_freq": 0,
        "cpu_min_count": 1,
        "ram_size_mb": 64.0,
        "storage_size_gb": 1.0
    },
    "command": {
        "binary": "python3",
        "args": ["/tmp/fos_helloworld.py"]
    },
    "hypervisor": "BARE",
    "migration_kind": "COLD",
    "storage": [],
```

(continues on next page)

(continued from previous page)

```
    "depends_on": [],
    "interfaces": [],
    "io_ports": [],
    "connection_points": []
}
```

Now if we suppose to have an Eclipse fog05 node in our localhost, we can use the Eclipse fog05 Python FIM API to register and deploy this component.

Let's imagine that our descriptor was saved in our `$HOME` directory, we can use this simple python script to deploy the "hello world" example

```
from fog05 import FIMAPI

def read_file(filepath):
    with open(filepath, 'r') as f:
        data = f.read()
    return data

n = '<our node id>'
api = FIMAPI()
desc = read_file('$HOME/fdu_helloworld.json')

fduD = api.fdu.onboard(desc)
print ('fdu_id : {}'.format(fduD.get_uuid()))
time.sleep(2)
inst_info = api.fdu.instantiate(fdu_id, n)
print ('Instance ID : {}'.format(inst_info.get_uuid()))

input('Press enter to terminate')

api.fdu.terminate(inst_info.get_uuid())

api.close()
exit(0)
```

We can save this file as `fos_deploy.py` and run it using `python3`

```
$ python3 fos_deploy.py
fdu_id: 6f52866c-0e69-4075-9ee1-b24c3b8a0969
Instance ID: 4ac683a6-7fee-4481-af2b-c7ca6c5bdf9c
Press enter to terminate
...
$
```

Before the termination you can run *c'at /tmp/fos_helloworld'* and you shold get something similar to

```
$ cat /tmp/fos_helloworld
Hello, world! It is 1571134486584032000
Hello, world! It is 1571134488586158000
Hello, world! It is 1571134490586510000
Hello, world! It is 1571134492586757000
Hello, world! It is 1571134494589501000
Hello, world! It is 1571134496589933000
```

CHAPTER 3

API

Eclipse fog05 Client provides the following APIs:

3.1 FIM Client API

Eclipse fog05 FIM Client API

3.1.1 FIMAPI

class fog05.fimapi.**FIMAPI**(*locator='127.0.0.1:7447'*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)

Class: FIMAPI

This class implements the API to interact with Eclipse fog05 FIM

Attributes

descriptor [Descriptor] Gives access to the descriptor API

node [Node] Gives access to the node API

plugin [Plugin] Gives access to the plugin API

network [Network] Gives access to the descriptor API

fdu [FDUAPI] Gives access to the FDU API

image [Image] Gives access to the image API

flavor [Flavor] Gives access to the flavor API

3.1.2 Descriptor

class fog05.fimapi.FIMAPI.**Descriptor**

Class: Descriptor

This class encapsulates API for descriptors

Methods

Type	Descriptor types
check(self, descriptor, descriptor_type)	Checks the given descriptor

class Type

Descriptor types

check(*self*, *descriptor*, *descriptor_type*)

Checks the given descriptor

Parameters

descriptor [dictionary] the descriptor to be checked

descriptor_type [API.Descriptor.Type] type of descriptor

Returns

bool

3.1.3 Node

class fog05.fimapi.FIMAPI.**Node**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)

Class: Node This class encapsulates API for Nodes

Methods

info(self, node_uuid)	Provides all information about the given node
list(self)	Gets all nodes in the current system/tenant
plugins(self, node_uuid)	Gets the list of plugins in the given node
search(self, search_dict)	Searches for nodes that satisfies the parameter
status(self, node_uuid)	Provides all status information about the given node,

info(*self*, *node_uuid*)

Provides all information about the given node

Parameters

node_uuid [string] UUID of the node

Returns

dictionary

list(*self*)

Gets all nodes in the current system/tenant

Returns

string list

plugins (*self*, *node_uuid*)
Gets the list of plugins in the given node

Parameters

node_uuid [string] UUID of the node

Returns

string list

search (*self*, *search_dict*)
Searches for nodes that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

Returns

string list

status (*self*, *node_uuid*)
Provides all status information about the given node,

Parameters

node_uuid [string] UUID of the node

Returns

dictionary

3.1.4 Plugin

class fog05.fimapi.FIMAPI.**Plugin**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)
Class: Plugin This class encapsulates API for Plugins

Methods

add(self, descriptor, node_uuid)	Adds the given plugin to the given node
info(self, plugin_uuid, node_uuid)	Gets information about the given plugin in the given node
remove(self, plugin_uuid, node_uuid)	Removes the given plugin from the given node
search(self, search_dict[, node_uuid])	Searches for plugin that satisfies the parameter

add (*self*, *descriptor*, *node_uuid*)
Adds the given plugin to the given node

Parameters

descriptor [dictionary] the plugin descriptor

node_uuid [string] UUID of the node

Returns

bool

info(*self*, *plugin_uuid*, *node_uuid*)
Gets information about the given plugin in the given node

Parameters

plugin_uuid [string] UUID of the plugin

node_uuid [string] UUID of the node

Returns

dictionary

remove(*self*, *plugin_uuid*, *node_uuid*)
Removes the given plugin from the given node

Parameters

plugin_uuid [string] UUID of the plugin

node_uuid [string] UUID of the node

Returns

bool

search(*self*, *search_dict*, *node_uuid=None*)
Searches for plugin that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

node_uuid [string] optional node UUID where search

Returns

string list

3.1.5 Network

class fog05.fimapi.FIMAPI.**Plugin**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)
Class: Plugin This class encapsulates API for Plugins

Methods

add(self, descriptor, node_uuid)	Adds the given plugin to the given node
info(self, plugin_uuid, node_uuid)	Gets information about the given plugin in the given node
remove(self, plugin_uuid, node_uuid)	Removes the given plugin from the given node
search(self, search_dict[, node_uuid])	Searches for plugin that satisfies the parameter

add(*self*, *descriptor*, *node_uuid*)
Adds the given plugin to the given node

Parameters

descriptor [dictionary] the plugin descriptor

node_uuid [string] UUID of the node

Returns

bool

info(*self*, *plugin_uuid*, *node_uuid*)
Gets information about the given plugin in the given node

Parameters

plugin_uuid [string] UUID of the plugin

node_uuid [string] UUID of the node

Returns

dictionary

remove(*self*, *plugin_uuid*, *node_uuid*)
Removes the given plugin from the given node

Parameters

plugin_uuid [string] UUID of the plugin

node_uuid [string] UUID of the node

Returns

bool

search(*self*, *search_dict*, *node_uuid=None*)
Searches for plugin that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

node_uuid [string] optional node UUID where search

Returns

string list

3.1.6 FDUAPI

class fog05.fimapi.FIMAPI.**FDUAPI**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)
Class: FDUAPI This class encapsulates API for FDUs

Methods

clean(self, instanceid[, wait])	Cleans the given instance

Continued on next page

Table 5 – continued from previous page

configure(self, instanceid[, wait])	Configures the given instance
define(self, fduid, node_uuid[, wait])	Defines the given fdu in the given node
get_nodes(self, fdu_uuid)	Gets all the node in which the given FDU is running
info(self, fdu_uuid)	Gets information about the given FDU from the catalog
instance_info(self, instanceid)	Gets information about the given instance
instance_list(self, fduid)	Gets all the instances of a given FDU
instantiate(self, fduid, nodeid[, wait])	Instantiates the given fdu in the given node
list(self)	Gets all the FDUs registered in the catalog
list_node(self, node_uuid)	Gets all the FDUs running in the given node
migrate(self, instanceid, destination_node_uuid)	Migrates the given instance
offload(self, fdu_uuid)	Removes the given FDU from the system catalog Needs at least one node in the system!
onboard(self, descriptor)	Registers an FDU descriptor in the system catalog Needs at least one node in the system!
pause(self, instanceid[, wait])	Pauses the given instance
resume(self, instanceid[, wait])	Resumes the given instance
search(self, search_dict[, node_uuid])	Searches for flavors that satisfies the parameter
start(self, instanceid[, wait])	Starts the given instance
stop(self, instanceid[, wait])	Stops the given instance
terminate(self, instanceid[, wait])	Terminates the given instance
undefine(self, instanceid)	Undefines the given instance

clean (*self*, *instanceid*, *wait=True*)
Cleans the given instance

Returns

string

configure (*self*, *instanceid*, *wait=True*)
Configures the given instance

Returns

string

define (*self*, *fduid*, *node_uuid*, *wait=True*)
Defines the given fdu in the given node

Instance UUID is system-wide unique

Parameters

fduid [string] UUID of the FDU

node_uuid [string] UUID of the node

wait [bool] optional, call will block until FDU is defined

returns

——-

InfraFDU

get_nodes (*self*, *fdu_uuid*)
Gets all the node in which the given FDU is running

Parameters

fdu_uuid [string] UUID of the FDU

Returns

string list

info(*self*, *fdu_uuid*)
Gets information about the given FDU from the catalog

Parameters

fdu_uuid [string] UUID of the FDU

Returns

FDU

instance_info(*self*, *instanceid*)
Gets information about the given instance

Parameters

instanceid [string] UUID of the instance

Returns

InfraFDU

instance_list(*self*, *fduid*)
Gets all the instances of a given FDU

Parameters

fduid [string] UUID of the FDU

Returns

dictionary {node_id: [instances list]}

instantiate(*self*, *fduid*, *nodeid*, *wait=True*)
Instantiates the given fdu in the given node

This functions calls: define, configure, start

Instance UUID is system-wide unique

Parameters

fduid [string] UUID of the FDU

node_uuid [string] UUID of the node

wait [bool] optional, call will block until FDU is defined

Returns

InfraFDU

list(*self*)
Gets all the FDUs registered in the catalog

Returns

string list

list_node(*self*, *node_uuid*)
Gets all the FDUs running in the given node

node_uuid [string] UUID of the node

Returns

string list

migrate(*self*, *instanceid*, *destination_node_uuid*, *wait=True*)
Migrates the given instance

Returns

string

offload(*self*, *fdu_uuid*)
Removes the given FDU from the system catalog Needs at least one node in the system!

Parameters

fdu_uuid [string] UUID of fdu

Returns

string

onboard(*self*, *descriptor*)
Registers an FDU descriptor in the system catalog Needs at least one node in the system!

Parameters

descriptor [FDU] FDU descriptor

Returns

FDU

pause(*self*, *instanceid*, *wait=True*)
Pauses the given instance

Returns

string

resume(*self*, *instanceid*, *wait=True*)
Resumes the given instance

Returns

string

search(*self*, *search_dict*, *node_uuid=None*)
Searches for flavors that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

node_uuid [string] optional node UUID where search

Returns

string list

start(*self*, *instanceid*, *wait=True*)
Starts the given instance

Returns

string

stop(*self*, *instanceid*, *wait=True*)
Stops the given instance

Returns

string

terminate(*self*, *instanceid*, *wait=True*)
Terminates the given instance

This function calls: stop, clean, undefine

Returns

string

undefine(*self*, *instanceid*)
Undefines the given instance

Returns

string

3.1.7 Image

class fog05.fimapi.FIMAPI.**Image**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)
Class: Image This class encapsulates API for Images

Methods

add(self, descriptor)	Registers an image in the system catalog Needs at least one not in the system
get(self, image_uuid)	Gets the information about the given image
list(self)	Gets all the images registered in the system catalog
remove(self, image_uuid)	Removes the given image from the system catalog Needs at least one not in the system
search(self, search_dict[, node_uuid])	Searches for images that satisfies the parameter

add(*self*, *descriptor*)
Registers an image in the system catalog Needs at least one not in the system

Parameters

descriptor [dictionary] image descriptor

Returns

string

get(*self*, *image_uuid*)
Gets the information about the given image

Parameters

image_uuid [string] UUID of image

Returns

dictionary

list (*self*)
Gets all the images registered in the system catalog

Returns

string list

remove (*self*, *image_uuid*)
Removes the given image from the system catalog Needs at least one not in the system

Parameters

image_uuid [string]

Returns

string

search (*self*, *search_dict*, *node_uuid=None*)
Searches for images that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

node_uuid [string] optional node UUID where search

Returns

string list

3.1.8 Flavor

class fog05.fimapi.FIMAPI.**Flavor**(*connector=None*, *sysid=<sphinx.ext.autodoc.importer._MockObject object>*, *tenantid=<sphinx.ext.autodoc.importer._MockObject object>*)
Class: Flavor This class encapsulates API for Flavors

Methods

add(self, descriptor)	Registers a flavor in the system catalog Needs at least one node in the system
get(self, flavor_uuid)	Gets information about the given flavor
list(self)	Gets all the flavors registered in the system catalog
remove(self, flavor_uuid)	Removes the given flavor from the system catalog Needs at least one node in the system
search(self, search_dict[, node_uuid])	Searches for flavors that satisfies the parameter

add (*self*, *descriptor*)
Registers a flavor in the system catalog Needs at least one node in the system

Parameters

descriptor [dictionary] flavor descriptor

Returns

string

get(*self*, *flavor_uuid*)
Gets information about the given flavor

Parameters

flavor_uuid [string] UUID of flavor

Returns

dictionary

list(*self*)
Gets all the flavors registered in the system catalog

Returns

string list

remove(*self*, *flavor_uuid*)
Removes the given flavor from the system catalog Needs at least one node in the system

Parameters

flavor_uuid [string] UUID of flavor

Returns

string

search(*self*, *search_dict*, *node_uuid=None*)
Searches for flavors that satisfies the parameter

Parameters

search_dict [dictionary] search parameters

node_uuid [string] optional node UUID where search

Returns

string list

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

T

U