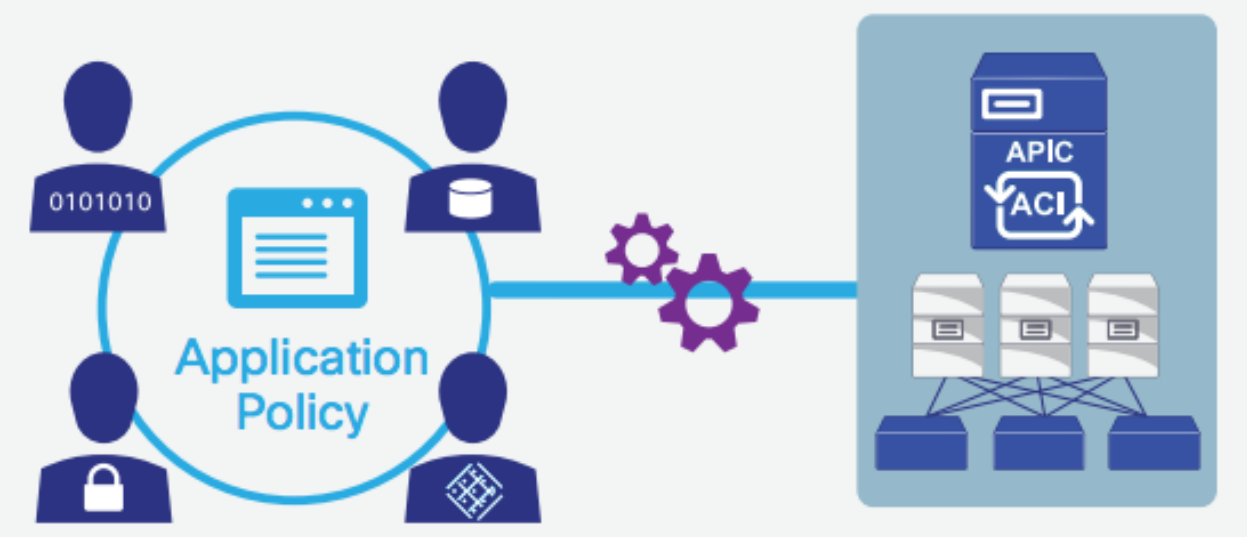
acitoolkit Documentation

Release 0.1

Cisco Systems, Inc.

May 05, 2017

Contents



Contents:

CHAPTER 1

Introduction

The Cisco ACI Fabric is configured using an abstract policy model on the Cisco Application Policy Infrastructure Controller (APIC). The APIC has a very rich and complete object model that is accessible through a programmatic REST API. The acitoolkit exposes a small subset of that model in a way that is meant to provide an introduction to the ACI concepts and allow users to get the most common workflows up and running as quickly as possible.

CHAPTER 2

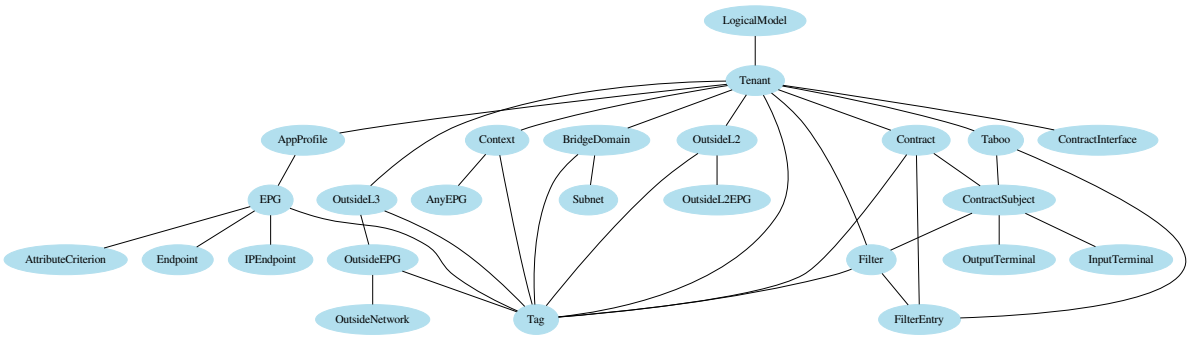
Object Model

The acitoolkit object model is divided into 3 sub-areas

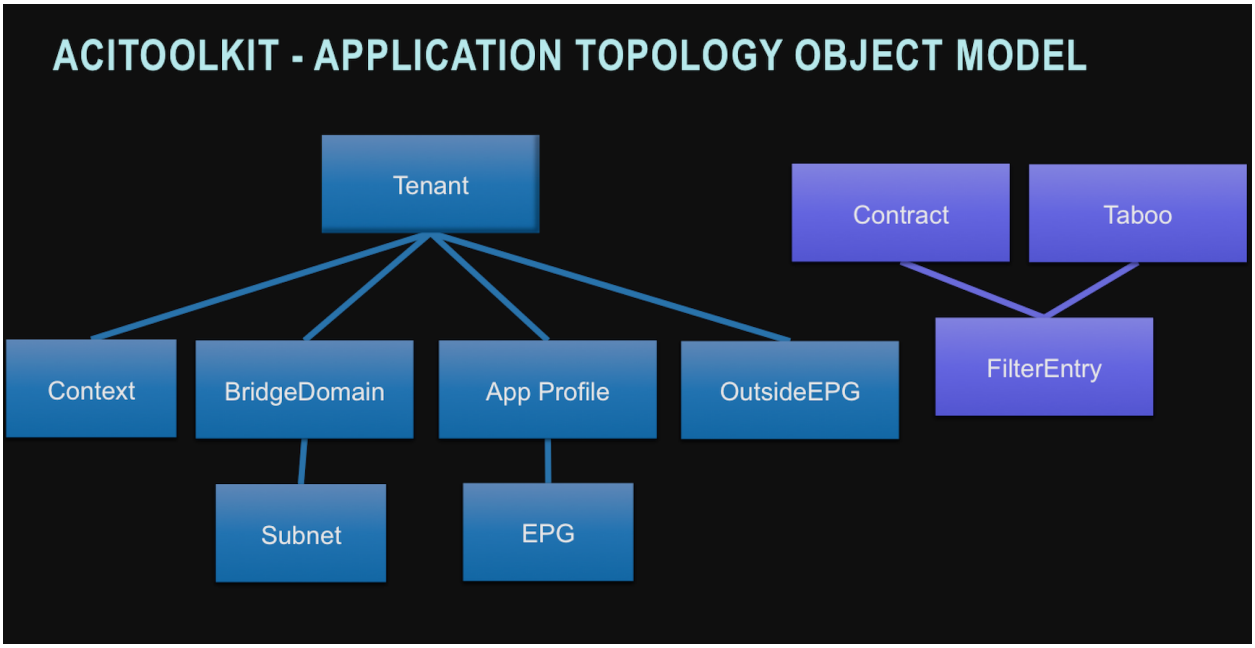
- Application Topology Object Model
- Interface Object Model
- Physical Topology Object Model

Application Topology

The acitoolkit defines the fabric configuration using a set of policies that describes the application logical topology. The following diagram shows the full logical topology diagram autogenerated from the source code.



Some of the key classes are shown and described below with the remainder described in the API Reference.



Tenant is the root class within the acitoolkit object model hierarchy. All of the application topology configuration occurs within a Tenant.

AppProfile is the Application Profile class. It contains the configuration for a given application.

EPG is the Endpoint Group class. This is the object for defining configuration that is applied when endpoints connect to the fabric.

Context is the class representing an L3 namespace (roughly, a traditional VRF in Cisco terminology).

BridgeDomain is the class representing an L2 forwarding domain (roughly, a traditional VLAN). It is associated with a single Context.

Subnet is the class representing an L3 subnet. It is associated with a single BridgeDomain.

OutsideEPG is the class representing an EPG that connects to the world outside the fabric. This is where routing protocols such as OSPF are enabled.

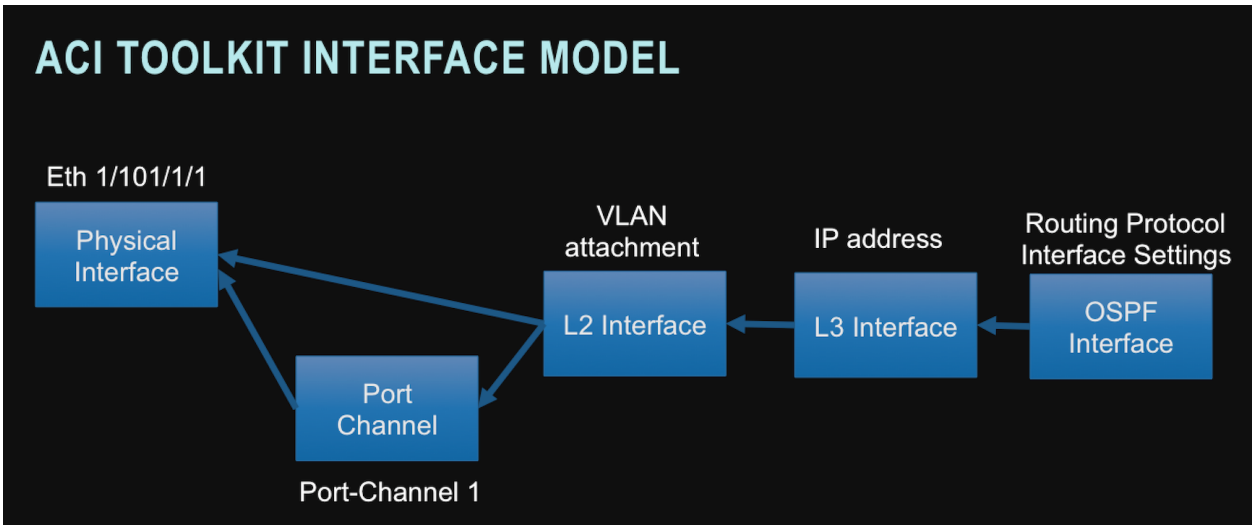
Contracts define the application network services being provided and consumed by EPGs. EPGs may provide and consume many contracts.

Taboos define the application network services that can never be provided or consumed by EPGs.

FilterEntry contained within either a Contract or a Taboo. Defines the traffic profile that the Contract or Taboo applies.

Interface Object Model

Interfaces provide the linkage between the application logical topology and the underlying physical network topology. The Interface set of classes are connected through a series of attachment relationships as shown in the class diagram below.



Interface class represents the **Physical Interfaces**. These are the objects that link the logical topology with the physical world. These objects represent the access ports on the leaf switches. These are the interfaces that the endpoints will physically attach.

Link aggregation

A *link aggregation* is a logical link layer interface composed of one or more physical interfaces. Commonly referred to as Etherchannel or PortChannel.

PortChannel class represents the logical aggregated ethernet port formed by Link Aggregation. This is done by creating a PortChannel instance and attaching one or more Interface instances to it. When interfaces belonging to 2 separate switches are assigned to the same PortChannel, this is referred to as a VPC or Virtual Port Channel. In the acitoolkit, VPCs are also represented by the PortChannel class.

L2Interface class represents the logical L2 network attachment on an Ethernet interface. In this case, the Ethernet interface could be an Interface class instance or PortChannel class instance as both are considered representations of link layer Ethernet interfaces.

Multiple logical L2 network attachments can occur on the same Ethernet interface. When this occurs, the L2Interface instances must use different encapsulation identifiers and/or different encapsulation types. The valid encapsulation types are:

- VLAN
- VXLAN
- NVGRE

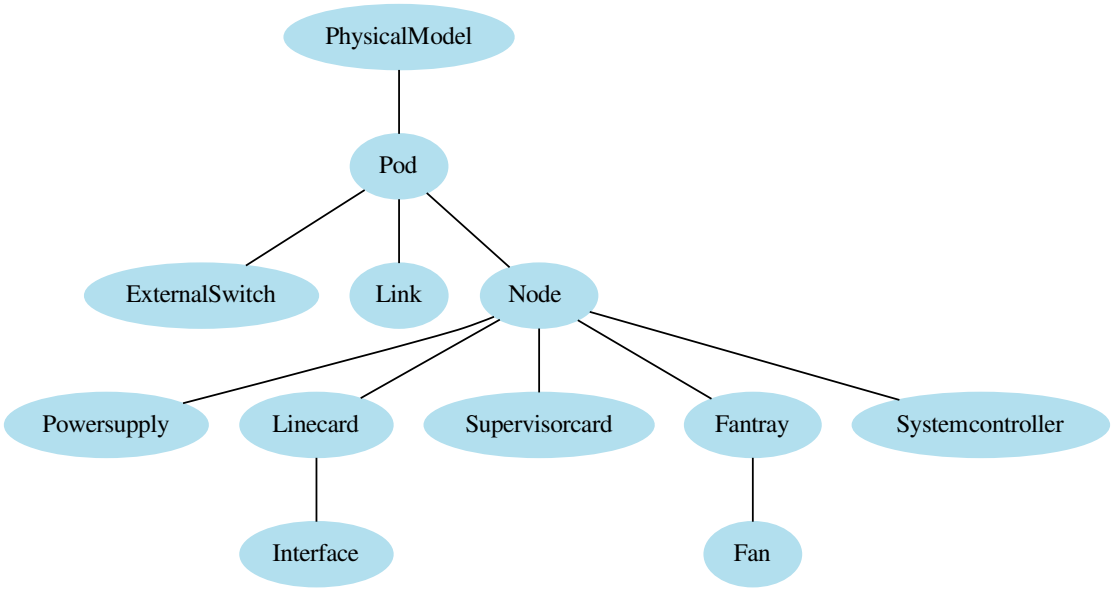
L3Interface class represents the logical L3 network attachment on an L2Interface. The L3Interface instance is where the IP address resides.

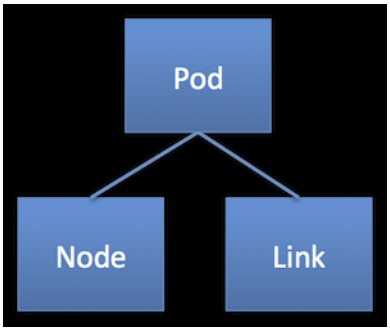
OSPFInterface class represents the logical router interface that routes from the L3Interface instance. It contains the OSPF-specific interface configuration.

Physical Topology

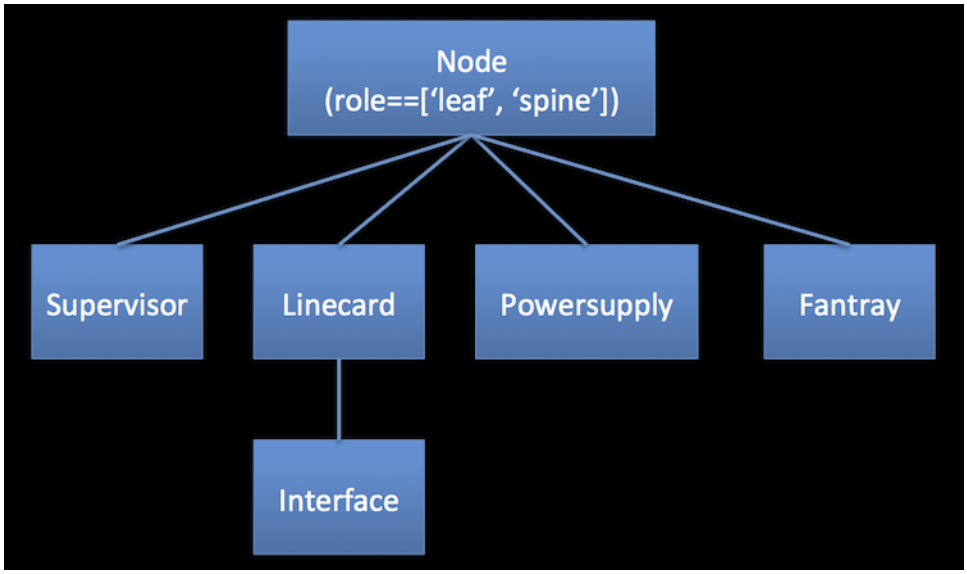
The acitoolkit defines the network topology using a set of objects that represent each of the components of the topology.

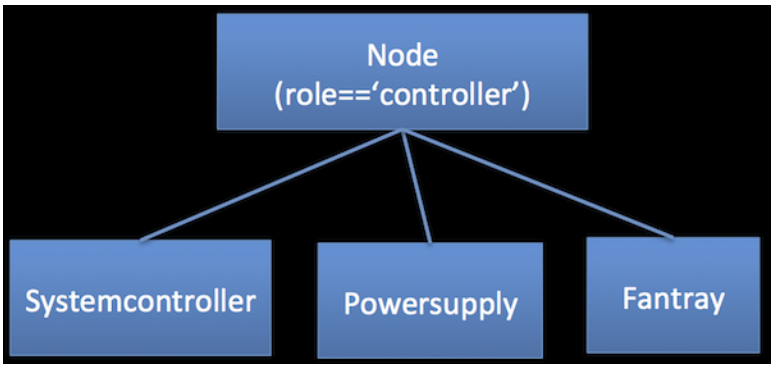
These objects are connected in a hierarchy according to the following diagrams.





The *acitoolkit.aciphysobject.Node* object is used to represent both switches and controllers. Which kind of *acitoolkit.aciphysobject.Node* can be determined by looking at the role, `Node.role`, attribute. Switches are Nodes with the `role` of 'leaf' or 'spine' and controllers are Nodes with the `role` of 'controller'. Switches are composed slightly different from controllers as shown in the following diagram.





Pod is the class for a physical Pod. *acitoolkit.aciphysobject.Pod* conatins of all the switches, links, and controllers that connected in the simple leaf-spine fat tree topology of the ACI fabric. It does not include the end-points or other devices that are attached to the ACI fabric.

Node is the class used to represent switches and controllers. What role the *acitoolkit.aciphysobject.Node* plays in the fabric can be determined by looking at the `role` attribute.

Link is the class representing links in the fabric. *acitoolkit.aciphysobject.Link* includes links between leaf and spine switches as well as links from leaf switches to controllers. Each link has two ends, the first and second end, in no particular order. This class has methods for retrieving the Switch, Linecard and Interface for each of the ends of the link.

Supervisorcard is the class representing the supervisor card in a switch. *acitoolkit.aciphysobject.Supervisorcard* would only be a child of a Node that has the role of 'leaf' or 'spine'. Each switch will have a supervisor including fixed configuration switches that may not have a obviously physically separate module that is a supervisor. The supervisor is where the primary software of the switch runs.

Linecard is the class representing a linecard in a switch. The *acitoolkit.aciphysobject.Linecard* is where all of the physical interfaces or ports are attached. In modular switches, the linecard is physically obvious, but even fixed configuration switches have a linecard where all the interfaces, ports, reside. A specific linecard in a switch is identified by its `slot_id` attribute. The `slot_id` of a linecard in a fixed configuration switch is always '1'.

Powersupply *acitoolkit.aciphysobject.Powersupply* is the class representing a power supply in a node.

Fantray *acitoolkit.aciphysobject.Fantray* is the class representing a fan tray in a node

Systemcontroller *acitoolkit.aciphysobject.Systemcontroller* is the class representing a system contoller of an APIC controller. This is the motherboard of the controller and is a good place to understand the version of software that is running in the controller.

Interface `acitoolkit.acitoolkit.Interface` described above.

CHAPTER 3

Monitor Policy

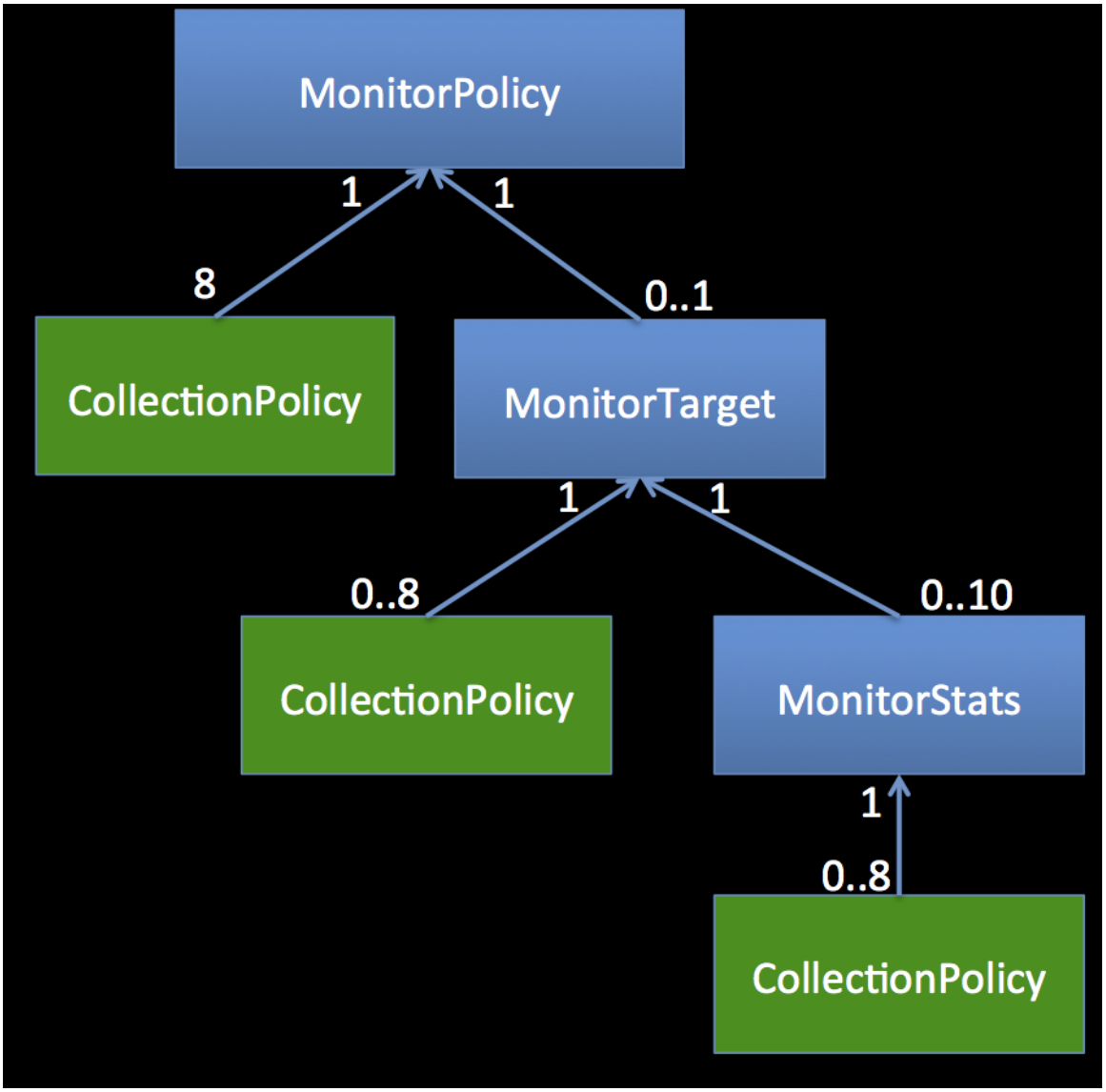
The monitor policy, *acitoolkit.acitoolkit.MonitorPolicy*, defines what statistical information is gathered and how long historical information is kept. It is also where events that are triggered by these stats are configured (to be supported).

Multiple monitoring policies can be defined and various APIC objects then reference the monitoring policy they are using. For example, a `l1PhysIf` object in the APIC has an attribute called `monPolDn` which is the distinguishing name of the monitoring policy that it references. In the toolkit, the `l1PhysIf` object is represented by the `acitoolkit.acitoolkit.Interface` class.

There are two types of monitoring policies:`fabric` and `access` and they are identified by the `policyType` attribute of the monitor policy. The `fabric` type is used to monitor ACI fabric interfaces while the `access` type is used to monitor ACI access or `infra` interfaces. The same class is used for both types of monitoring policies in the acitoolkit.

The monitoring policy is a hierarchical policy consisting of monitor policy class, *acitoolkit.acitoolkit.MonitorPolicy*, at the top with the following classes below it: *acitoolkit.acitoolkit.MonitorTarget*, *acitoolkit.acitoolkit.MonitorStats*, and *acitoolkit.acitoolkit.CollectionPolicy*.

The following diagram shows their relationship.



CollectionPolicy is where the actual collection policy is defined. What it applies to is determined by where it is in the monitoring policy hierarchy. The three most important attributes of the collection policy are `adminState`, `granularity` and `retention`. Additional attributes are a `name` and `description` which are optional and can be set using the `setName(<name>)` and `setDescription(<description>)` methods respectively.

The `adminState` attribute can be `enabled`, `disabled` or `inherited`. If `enabled`, that granularity of statistics will be gathered. If `disabled`, that granularity of statistics will not be gathered and *neither will any larger granularities*. This is because the statistics gathered at one granularity are then rolled up into the larger granularity. If you don't gather the finer one, then there is no data to roll up to the coarser one.

If the `adminState` is set to `inherited`, the current object does not determine the `adminState`. Instead, the `adminState` of the collection policy of the next higher level in the monitoring policy hierarchy will be used. This means that the `adminState` at the highest level of the monitoring policy *cannot* be set to `inherited` because there is no higher level to inherit from.

The `granularity` attribute can have one of the following values:

Value	Meaning
5min	5 minutes
15min	15 minutes
1h	1 hour
1d	1 day
1w	1 week
1mo	1 month
1qtr	1 quarter
1year	1 year

This is the time interval over which the stats are initially gathered and the interval for which they are kept.

For example, if the granularity is `15min`, then the cumulative stats for that granularity will start at 0 at the beginning of the 15 minute interval and will accumulate during the interval. At the end of the interval, the final values will be moved to the historical statistics if the `retention` attribute is so configured. The rate statistics will be the rate during the 15 minute interval and the rate averages will be the average rate during the 15 minute interval.

Statistics are only kept for granularities that have an adminState of `enabled` either explicitly or through inheritance and no finer (smaller) granularities are `disabled`.

The `retention` attribute determines how long historical data at a given granularity is kept. It can have one of the following values:

Value	Meaning
none	Do not keep historical data
inherited	Use the policy from the next higher level of hierarchy
5min	5 minutes
15min	15 minutes
1h	1 hour
1d	1 day
1w	1 week
10d	10 days
1mo	1 month
1qtr	1 quarter
1year	1 year
2year	2 years
3year	3 years

It does not make any sense to have a retention period that is less than the granularity, however this is not checked for in the acitoolkit.

MonitorStats sets the scope for any collection policy under it. The scope here is a family of statistics. The possible scope values are as follows:

Value	Description
egrBytes	Egress bytes
egrPkts	Egress packets
egrTotal	Egress total
egrDropPkts	Egress drop packets
ingrBytes	Ingress bytes
ingrPkts	Ingress packets
ingrTotal	Ingress total
ingrDropPkts	Ingress drop packets
ingrUnkBytes	Ingress unknown bytes
ingrUnkPkts	Ingress unknown packets

A more detailed description of the statistics can be found here.

The collection policies under the `MonitorStats` object determine the default collection policy for the set of statistics selected by the above scope.

Other attributes of the `MonitorStats` class are `name` and `description` which can be set with the `setName(<name>)` and `setDescription(<description>)` methods respectively. Setting these attributes is optional.

MonitorTarget sets the scope to a particular APIC target object for all of the collections policies below it. Currently, there is only one APIC target object type supported and that is 'l1PhysIf'. The `scope` attribute is where the target type is stored. Support for additional target objects will be added as required. The `scope` attributed is initialized when the MonitorTarget is created and cannot be changed.

Other attributes of the `MonitorStats` class are `name` and `description` which can be set with the `setName(<name>)` and `setDescription(<description>)` methods respectively. Setting these attributes is optional.

MonitorPolicy is the root of the monitor policy hierarchy. This object must have `name` and `policyType` attribute. The `policyType` must be either `fabric` or `access` and the name must be unique for each `policyType`.

The monitor policy will be referenced by its `policyType` and `name` by individual APIC objects.

The monitor policy contains the default collection policies as well as any `MonitorTarget` objects that specify a more specific scope.

The monitor policy must contain a `CollectionPolicy` for each granularity and the `adminState` and `retention` attributes of the `CollectionPolicy` cannot be `inherited` because they are at the top of the inheritance tree. When a MonitorPolicy object is created, it will be initialized with the appropriate `CollectionPolicy` objects, which, in turn, will be set to a default administrative state of `disabled`. This means that these polies *must* be overwritten if stats should be collected. They can either be explicitly replaced with the `add_collection_policy(<CollectionPolicy object>)` method, or implicitly replaced by more specific collection policies in the inheritance hierarchy.

Policy Resolution

The ultimate policy that is applied to any counter is determined by walking through the monitoring policy from the top down. The collection policy at each level of the hierarchy determines how statistics will be kept for those statistics that are *in scope*.

For example, the collection policy for each granularity is specified at the top of the hierarchy under the MonitorPolicy object. These collection policies will apply to all statistics unless overwritten by a more specific policy under a MonitorTarget object.

If there is a MonitorTarget object, it will set the scope for the monitoring policy to be more specific for the collection policies under it. Initially, the only target supported is 'l1PhyIf' which is for an `Interface` object. Any collection policies under this `MonitorTarget` will override the corresponding collection policy under the `MonitorPolicy` parent object. It is possible that there are no collection policies specified at this level.

If there are `MonitorStats` objects under the `MonitorTarget` object, they will set the scope to be even more specific for the collection policies under them. Each `MonitorStats` object can have under it collection policies for any of the granularities.

CHAPTER 4

Tutorial

This document is meant to give a tutorial-like overview of the acitoolkit package.

Setting up the environment

This tutorial will walk you through installing the acitoolkit using the sources so that you will be able to edit the samples and even the acitoolkit source code if so desired.

Download

First, we must download the acitoolkit. This is best done using git, but can be done by downloading the package as a zip.

If you have git installed, clone the repository using the following command:

```
git clone https://github.com/datacenter/acitoolkit.git
```

If git is not installed, you can download the acitoolkit as a zip file instead.:

```
wget https://github.com/datacenter/acitoolkit/archive/master.zip
unzip master.zip
```

Install

Note

The directory may be named `acitoolkit-master` if downloaded as a zip file.

Next, cd into the created directory

```
cd acitoolkit
```

and install the acitoolkit

```
python setup.py install
```

Note that when installing on Mac or Linux, you will likely need to run this as administrator so preface the command with the `sudo` keyword as follows:

```
sudo python setup.py install
```

If you plan on modifying the actual toolkit files, you should install the developer environment that will link the package installation to your development directory. Do this instead of the install option above.

python setup.py develop

Common Installation Errors

Missing development packages

Some of the dependencies may require that the python development environment be installed. This package is usually called `python-dev` or `python-develop`. This is usually the case when you see an error message referring to a missing file such as `Python.h: No such file or directory`.

In Ubuntu, you would install this package by `sudo apt-get install python-dev`

Pre-installed Packages

The `acitoolkit` can be downloaded pre-configured as a virtual machine.

Virtual Machine for VMware Environments

A pre-installed virtual machine in the form of a OVA file for VMware hypervisors can be found in the link below:

ACI Toolkit OVA

The virtual machine is configured with the following parameters:

```
Username: acitoolkit
Password: acitoolkit
Operating System: Ubuntu 16.04.2
Processor Cores: 1
Memory: 1GB
```

The `acitoolkit` and necessary packages are already installed. However, given the pace of change in datacenter networking, there most likely have been changes since the VM was created. Luckily, the VM can be updated to the latest version by entering the following command:

```
sudo ~/install
```

This command be re-run any time to get the latest updates.

Building a Simple Tenant Configuration

The following section will walk you through the implementation of the `tutorial.py` file found in the `/samples/` directory. This code will create a minimal configuration that will configure 2 interfaces on the fabric to be on the same network so that they can communicate. This code can be executed with the following command from within the `/samples/` directory:

```
python tutorial.py
```

Configuration Object Definition

Imports

The first part of the tutorial program consists of the `import` statement. The `acitoolkit` module from the acitoolkit package is imported.

```
from acitoolkit.acitoolkit import *
```

The `acitoolkit` module within the acitoolkit package provides access to all of the acitoolkit configuration.

Tenant Creation

All of the configuration will be created within a single tenant named `tutorial`. This is done by creating an instance of the Tenant class and passing it a string containing the tenant name.

```
tenant = Tenant('tutorial')
```

Application Profile

The Application Profile contains all of the Endpoint Groups representing the application. The next line of code creates the application profile. It does this by creating an instance of the `AppProfile` class and passing it a string containing the Application Profile name and the `Tenant` object that this `AppProfile` will belong.

```
app = AppProfile('myapp', tenant)
```

Note that many of the objects within the acitoolkit are created in this way, namely with a name and a parent object. The parent object must be an instance of this class's parent class according to the acitoolkit object model. The parent class of `AppProfile` is `Tenant`.

Endpoint Group

The Endpoint Group provides the policy based configuration for Endpoints that are members of the Endpoint Group. This is represented by the `EPG` class. In this case, we create an `EPG` with the name *myepg* and pass the `AppProfile` that we created to be the parent object.

```
epg = EPG('myepg', app)
```

Context and Bridge Domain

We also need an L3 namespace and L2 forwarding domain so we create the `Context` and `BridgeDomain` in the same manner as we did for the previous objects. For both of these classes, the parent class is Tenant.

```
context = Context('myvrf', tenant)
bd = BridgeDomain('mybd', tenant)
```

We then associate the `BridgeDomain` instance with the `Context` instance. This indicates that this `BridgeDomain` exists within this `Context`.

```
bd.add_context(context)
```

The `EPG` is then associated with the `BridgeDomain` that we created.

```
epg.add_bd(bd)
```

Associating the tenant configuration with the network

At this point, the tenant configuration is complete. However, it is not bound to the physical network yet so let's connect the EPG to 2 interfaces.

Physical Interfaces

First, we must create objects to represent the physical interfaces using the `Interface` class. Interface objects are named using interface type, pod, node (switch), module (linecard), and port names. In this case, the interface type is `'eth'` for ethernet and the interfaces are located in pod 1 on leaf switch 101 in module 1 within ports 15 and 16.

```
if1 = Interface('eth', '1', '101', '1', '15')
if2 = Interface('eth', '1', '101', '1', '16')
```

VLANs

In order to allow multiple EPGs to connect to the same interface, the ACI fabric uses network virtualization technologies such as VLAN, VXLAN, and NVGRE to keep the traffic isolated. In this case, we chose to use VLAN since it is the most ubiquitous and we chose to use the same encapsulation on both physical interfaces, namely VLAN 5.

The `L2Interface class` represents the virtual L2 network interface. In this case, this is the VLAN attached to a given physical interface. This is the interface where L2 protocols (such as spanning tree in traditional networks) run. Link layer protocols such as LLDP run directly on the physical interface independent of VLANs.

We create the `L2Interface` and pass a name `vlan5_on_if1`, the encapsulation technology `vlan`, and the virtual network identifier `5` as part of the contructor.

```
vlan5_on_if1 = L2Interface('vlan5_on_if1', 'vlan', '5')
```

We next associate this `L2Interface` to the desired physical interface.

```
vlan5_on_if1.attach(if1)
```

And we repeat for the second physical interface.

```
vlan5_on_if2 = L2Interface('vlan5_on_if2', 'vlan', '5')
vlan5_on_if2.attach(if2)
```

Now, we simply associate the `EPG` with the `L2Interface` objects that we created.

```
epg.attach(vlan5_on_if1)
epg.attach(vlan5_on_if2)
```

Deploying to the APIC

At this point, the entire configuration is done and all that is left is connecting to the APIC and deploying the configuration.

APIC Login Credentials

The APIC login credentials are retrieved using an instance of the `Credentials` class. This class provides a convenient mechanism to retrieve credentials and is used by a variety of toolkit applications.

The Credentials object is instantiated with a string describing the type of credentials desired and a description string.

```
description = 'acitoolkit tutorial application'
creds = Credentials('apic', description)
```

The command line is also extensible through the `add_argument` function. This is the same `add_argument` function provided by the standard `argparse` python package. In this tutorial, we extend the command line options with a delete flag so that we can clean up the configuration afterwards.

```
creds.add_argument('--delete', action='store_true',
                   help='Delete the configuration from the APIC')
```

Retrieving the credentials is done by calling the `get` function.

```
args = creds.get()
```

The `apic` set of credential variables consist of the `username`, `password`, and `URL` of the APIC. The `Credentials` class allow the credentials to be provided in a number of formats and is taken in the following priority order

- Command line options
- Configuration file called `credentials.py`
- Environment variables
- Interactively querying the user

A search will be performed for each credential individually so that different methods can be used at the same time. For example, the username and URL can be passed as Command Line Options and the password can be collected by querying the user directly. For this tutorial, we will query the user directly.

APIC Login

Next, we log into the APIC. This is done through the `Session` class. We create an instance and pass it the login credentials, namely the `URL`, `LOGIN`, and `PASSWORD`.

```
session = Session(URL, LOGIN, PASSWORD)
```

We also initiate the actual login.

```
session.login()
```

Once the login is complete, we can now send our configuration to the APIC. This is done by calling the `Session` object with the `push_to_apic` function that requires a URL and the JSON data to send to the APIC. All of the configuration for the application topology is collected under the `Tenant`. In order to get the URL to use and the JSON for our configuration, we simply call the `Tenant` instance with `get_url` and `get_json` respectively.

```
resp = session.push_to_apic(tenant.get_url(), data=tenant.get_json())
```

The `push_to_apic` call returns an object. This object is an instance of the `Response` class from the popular requests library which provides a rich set of return codes and status. Here, we simply check that the call was successful.

```
if resp.ok:
   print 'Success'
```

APIC Login (Certificate based)

The APIC REST API also supports authentication using certificates.

Once setup, it is a more simple and secure form of authentication, with each request being uniquely signed. Additionally, login timeout issues are removed. An important point to note is that websockets (events) are not supported by the APIC when using certficate authentication, so the corresponding acitoolkit functionality will be disabled.

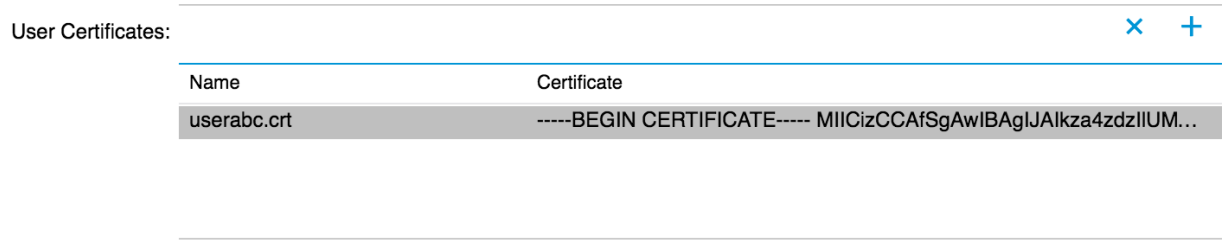
As a prerequisite you must have created a private key and public certificate and attached the certificate to the desired user using the APIC Web UI.

Creating a certificate session using the acitoolkit is simple:

1. Use OpenSSL to generate a X.509 certificate and private key.

```
openssl req -new -newkey rsa:1024 -days 36500 -nodes -x509 -keyout userabc.key -out␣
↪userabc.crt -subj '/CN=User ABC/O=Cisco Systems/C=US'
```

2. Upload the generated certificate `userabc.crt` to the user via the APIC



3. Certificate authentication has an extra dependency, not installed by default, which can be easily installed using pip

```
pip install pyopenssl
```

4. Create a certificate based authentication session

```
# Generic
session = Session(URL, LOGIN, cert_name=CERT_NAME, key=KEY)

# Example
session = Session('https://1.1.1.1', 'userabc', cert_name='userabc.crt', key='userabc.
↪key')
```

Note: If using the acitoolkit from the context of an APIC App Center app, make sure to pass the extra parameter `appcenter_user=True`. App Center apps are provided a user that belongs to a different class of users. The login and cert_name for App Center users are both in the form of `vendor_appId`. App Center users support certificate subsciptions through a special requestAppToken api. To use subscriptions with an App Center user, you must explicitly call the `login()` method which acquires and maintains the App user token. Disable App center subscriptions by setting the parameter `subscription_enabled=False`.

You do not need to explicitly call the `login()` method when using certificate authentication.

After this point, you can continue to use all of the acitoolkit methods to get and push configuration from the APIC securely and without logging in.

Displaying the JSON Configuration

At this point, we're done ! The configuration has been sent to the APIC. Congratulations, you just programmed a datacenter fabric ! You should be able to see your new tenant `tutorial` within the APIC GUI with its new EPG and static path bindings.

The next few lines in the `tutorial.py` file simply print what was sent to the APIC. You can use this to manually edit the JSON if you wish to access the richer API on the APIC that the acitoolkit does not expose.

```
print 'Pushed the following JSON to the APIC'
print 'URL:', tenant.get_url()
print 'JSON:', tenant.get_json()
```

Removing the tenant configuration

You might have noticed that we jumped over 2 lines of the tutorial code, specifically the following lines.

```
if args.delete:
    tenant.mark_as_deleted()
```

The `args.delete` is set if the `--delete` command line option is given. Calling the `mark_as_deleted` function will cause the tenant to be deleted from the APIC when the configuration is pushed. It should be noted that deleting the tenant will cause all of the configuration for the tenant to be deleted. This will allow us to run the tutorial and then run it again to delete the configuration by executing the following commands.:

```
python tutorial.py
python tutorial.py --delete
```

The first command will push the configuration to the APIC and the second command will delete the configuration leaving you where we started.

Getting APIC objects

With the acitoolkit, it is possible to get objects from the APIC either on demand or through object event subscriptions. In most cases, getting the objects on demand will be sufficient. However, in cases where action needs to be taken immediately or to avoid frequent polling of the APIC, event subscriptions can be used.

Objects on demand

Getting objects on demand is fairly straightforward. Each class that allows getting objects from the APIC has a `get` class method. This method will return a list of objects belonging to that class type and will be retrieved from the APIC immediately upon calling.

Since the acitoolkit can be used to control multiple APICs at the same time, the `Session` class instance representing the connection to the desired APIC is also passed.

An example is shown in the code snippet below.:

```
tenants = Tenant.get(session)
for tenant in tenants:
    print tenant.name
```

Event subscriptions

Event subscriptions allow immediate notification when an object is created, modified, or deleted. Events will be received only for classes or instances that are subscribed.

Class subscriptions

To create a class subscription, the class method `subscribe` is called on the desired class along with the appropriate `Session` class instance. This is shown in the code snippet below using the `Tenant` class as the example.:

```
Tenant.subscribe(session)
```

To check an event has arrived, the method `has_event` can be called on the subscribed class.:

```
Tenant.has_event(session)
```

If there is an event waiting, this will return `True`.

Note: While this may look like it requires polling the APIC, it is actually just checking a local event receive queue. This event queue is populated by a separate thread receiving events from the APIC. Thus, calling `has_event` will not result in additional communication with the APIC so that this call can be run in a tight loop with minimal overhead and/or spun into a seperate thread if desired.

To retrieve the event, a call is made to the `get_event` method as shown below.:

```
event = Tenant.get_event(session)
```

This will return a instance of the object with the appropriate settings indicating the change. For instance, if the `Tenant` named *Bob* is deleted, the event will return a `Tenant` instance with the name set to *Bob* and it will be marked as *deleted*.

To no longer receive events for this particular class, the class method `unsubscribe` can be called. This will cause the subscription to be removed from the APIC.:

```
Tenant.unsubscribe(session)
```

Under the covers, the event subscriptions use a web socket to communicate with the APIC to receive the events. The events are then collected by a thread and placed into an event queue that is then queried by user code. Event subscriptions are refreshed automatically by the toolkit using a separate thread.

Instance subscriptions

Instance subscriptions are the same as class subscriptions except that the events are limited to only that particular object instance such as:

```
bob = Tenant('bob')
bob.subscribe(session)
bob.has_event(session)
event = bob.get_event(session)
```

A more useful example would be the following code which will wait for an event for the instance of `Tenant` with the name *Bob* and then print a message if the instance was deleted.:

```
bob = Tenant('Bob')
bob.subscribe(session)
while True:
    if bob.has_event(session):
        bobs_event = bob.get_event(session)
        if bobs_event.is_deleted():
            print 'Bob was deleted'
```

CHAPTER 5

Statistics

This documents gives an overview of the statistics.

Statistics

Statistics are gathered at each Interface according to the monitoring policy referenced by that Interface. See *Monitor Policy* for details of the monitoring policy.

This section describes the statistic counters themselves.

Statistic Families

The statistics are broken into multiple statistic families and each family consists of a set of specific counters, rate values, and timestamps.

An enumerated list of statistic families is found in the MonitorStats class and can be found as follows:

```
>>>import acitoolkit.acitoolkit as ACI
>>>statistic_family_list = ACI.MonitorStats.statsFamilyEnum
>>>for statistic_family in statistic_family_list:
...  print statistic_family
```

The families are as follows:

- *egrbytes*
- *egrPkts*
- *egrTotal*
- *egrDropPkts*
- *ingrBytes*
- *ingrPkts*

- *ingrTotal*
- *ingrDropPkts*
- *ingrUnkBytes*
- *ingrUnkPkts*
- *ingrStorm*

Accessing Stats

Each statistic family is described in detail below and can be accessed via the `stats` object contained in the `Interface` object.

You first use the get() method to read the stats from the APIC controller.:

```
stats = interface.stats.get()
```

This will return a data structure that will allow each counter to be referenced by its name (see list above), a granularity, and an epoch number in the following manner:

```
counter = stats[<stats_family>][<granularity>][<epoch>][<counter_name>]
```

For example, if you wanted to show the per day total of ingress, unicast packets from the previous day you would do the following:

```
stats = interface.stats.get()
print stats['ingrPkts']['1h'][1]['unicastPer']
```

The specific counter names can be found at *Statistics Detail*.

Each counter family has an interval start and end value as well which can be used to understand exactly when the counters were gathered.:

```
print 'start', interface.stats['ingrPkts']['1h'][1]['intervalStart']
print 'end', interface.stats['ingrPkts']['1h'][1]['intervalEnd']
```

One thing to note about accessing the stats is that if a particular counter is not currently being kept by the APIC controller, that particular counter will not be returned by the get() method. This means that you should either test for its existence before accessing it, or use the standard python dictionary get method to return a default value that your code can handle:

```
print stats['ingrPkts']['1h'][1].get('unicastPer',0)
```

A typical example of counters that may not exist would be for an epoch that is not being retained or a granularity that is not be gathered.

One issue with the above is that some counters are floating point, some are integers and some are timestamps. Returning a default of zero can lead to inconsistent formatting. To work around this problem use the `retrieve()` method that will return the coutner value or a default value that is consistent. The format of the retrive method is as follows:

```
interface.stats.retrieve(<stats_family>,<granularity>,<epoch>,<counter_name>)
```

The get() method will load the counter values and then they are accessed by the retrieve method as follows:

```
interface.stats.get()
print interface.stats.retrieve('ingrPkts','1h',1,'unicastPer')
```

Note that the result of the `get()` method was not used. It did cause a read of the stats from the APIC which are then stored in the `interface.stats` object. After that, the `interface.stats.retrieve()` method will access those previously read counters. The `retrieve()` method will not refresh the counters.

aci-show-interface-stats.py

The interface stats can also be accessed via the simple python script `aci-show-interface-stats.py`. This script has a couple of display options to customize the output.

A simple run of the script will display each interface in the network and a couple of selected stats for each:

```
python aci-show-interface-stats.py
```

The default display is for the `5min` granularity and the current, i.e. 0, epoch. An alternative granularity can be selected with the `-granularity` command line option.:

```
python aci-show-interface-stats.py -granularity 1h
```

The epoch can be specified with the `-epoch` option.:

```
python aci-show-interface-stats.py -granularity 1h -epoch 3
```

A specific interface can be specified with the `-interface` option. This might be useful if there are a large number of interfaces.:

```
python aci-show-interface-stats.py -g 1h -e 3 -interface 1/201/1/1
```

Note that we are also showing the abbreviated form of the other command line options. The above will show stats for pod 1, switch 201, slot 1, port 1.

If all of the stats for a given interface are desired, the `-full` option should be used.

    python aci-show-interface-stats.py -g 1h -e 3 -i 1/201/1/1 -full

This last option will show only those stats that have been collected according to the monitoring policy. Also, note that this last option only works when the `-interface` option is also used.

Granularity

The `<granularity>`, also called "interval", must be one of:

- 5min
- 15min
- 1h
- 1d
- 1w
- 1mo
- 1qtr
- 1year

An enumerated list of granularities is found in the CollectionPolicy class and can be found as follows:

```
>>>import acitoolkit.acitoolkit as ACI
>>>granularity_list = ACI.CollectionPolicy.granularityEnum
>>>for granularity in granularity_list:
...  print granularity
```

Epoch

The <epoch> is an integer representing which set of historical stats you want to reference. Epoch 0 is the current epoch which has not yet completed. Epoch 1 is the most recent one and so on. The length of each epoch is determined by the granularity.

The number of epochs available will be determined by the retention policy and granularity specified in the monitoring policy and how long they have been in place.

For example, if the monitoring policy for a particular statistics family has a granularity of 5min and a retention policy of 1h and it has been in place for more than one hour, then there will be a total of 13 epochs, 0 through 12. Epoch 0 will be the one currently active. Epoch 1 will be for the previous 5 minute interval. Epoch 2 will be for the 5 minute interval previous to epoch 1 and so on. At the beginning of the current Epoch, the values in Epoch 0 will be distorted because they are only for a fraction of that epoch (potentially a zero fraction) and the other 12 will represent an hour of history.

Update Frequency for current epoch

The current epoch, epoch 0, will be updated as it occurs, i.e. in near real-time. The interval that it updates depends on the epoch, or interval, granularity.

Granularity	Update frequency
5min	Every 10 seconds
15min	Every 5 minutes
1h	Every 15 minutes
1d	Every hour
1w	Every day
1mo	Every day
1qtr	Every day
1year	Every day

Statistics Detail

The following are details about each of the counters

egrBytes

Name	Type	Description
floodavg	integer	**Egress flood bytes average**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
floodCum	integer	**Egress flood bytes cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
floodMax	integer	**Egress flood bytes maximum**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
floodMin	integer	**Egress flood bytes minimum**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

egrTotal

Name	Type	Description
bytesAvg	integer	**Egress bytes average** This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
bytesCum	integer	**Egress bytes cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
bytesMax	integer	**Egress bytes maximum**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
bytesMin	integer	**Egress bytes minimum**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

ingrBytes

Name	Type	Description
floodavg	integer	**Ingress flood bytes average**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
floodCum	integer	**Ingress flood bytes cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
floodMax	integer	**Ingress flood bytes maximum**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
floodMin	integer	**Ingress flood bytes minimum**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

ingrTotal

Name	Type	Description
bytesAvg	integer	**Ingress bytes average** This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
bytesCum	integer	**Ingress bytes cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
bytesMax	integer	**Ingress bytes maximum**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
bytesMin	integer	**Ingress bytes minimum**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

egrPkts

Name	Type	Description
floodAvg	integer	**Egress flood average packets**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
floodCum	integer	**Egress flood cumulative packets**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
floodMax	integer	**Egress flood maximum packets**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
floodMin	integer	**Egress flood minimum packets**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

ingrPkts

Name	Type	Description
floodAvg	integer	**Ingress flood average packets**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
floodCum	integer	**Ingress flood cumulative packets**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
floodMax	integer	**Ingress flood maximum packets**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
floodMin	integer	**Ingress flood minimum packets**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For

egrDropPkts

Name	Type	Description
afdWredAvg	integer	**Egress packets dropped due to AFD or WRED**. This is a count of the packets dropped due to whichever active queue managment mechanism is running in the switch, either AFD or WRED. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
afdWredCum	integer	**Egress packets dropped cumulative due to AFD or WRED**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
afdWredMax	integer	**Egress packets dropped by AFD or WRED maximum value read**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
afdWredMin	integer	**Egress packets dropped by AFD or WRED minimum value read**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For example, if the value of the first reading is 3

ingrDropPkts

Name	Type	Description
bufferAvg	integer	**Ingress packets dropped due to buffer full**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
bufferCum	integer	**Ingress packets dropped cumulative due to buffer full**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
bufferMax	integer	**Ingress packets dropped due to buffer full max read**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
bufferMin	integer	**Ingress packets dropped due to buffer full min read**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 2, the previous value is overwritten with 2. If the third reading is larger than 2, the value remains at 2.

ingrUnkBytes

Name	Type	Description
unclassifiedAvg	integer	**Ingress unclassified bytes average**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
unclassifiedCum	integer	**Ingress unclassified bytes cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
unclassifiedMax	integer	**Ingress unclassified bytes max value read**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
unclassifiedMin	integer	**Ingress unclassified bytes min value read**. This is the smallest value read by the counter during the collection interval. This value is only overwritten if the most current value is smaller than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 2, the previous value is overwritten with 2. If the third reading is larger than 2, the value remains at 2.

ingrStorm

Name	Type	Description
dropBytesAvg	integer	**Ingress ave bytes dropped due to storm control**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
dropBytesCum	integer	**Ingress cum bytes dropped due to storm control** The total sum of the values read. Note that this value continues through each interval without resetting to zero.
dropBytesMax	integer	**Ingress max bytes dropped due to storm control**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
dropBytesMin	integer	**Ingress min bytes dropped due to storm control** This is the smallest value read by the counter during the collection interval. This value is

ingrUnkPkts

Name	Type	Description
unclassifiedAvg	integer	**Ingress unclassified packets average**. This is the average value read by the counter during the collection interval. Note that this value resets to 0 at the beginning of each interval.
unclassifiedCum	integer	**Ingress unclassified packets cumulative**. The total sum of the values read. Note that this value continues through each interval without resetting to zero.
unclassifiedMax	integer	**Ingress unclassified packets max value read**. This is the largest value read by the counter during the collection interval. This value is only overwritten if the most current value is larger than the previous value. For example, if the value of the first reading is 3 and the value of the second reading is 4, the previous value is overwritten with 4. If the third reading is smaller than 4, the value remains at 4. Note that this value resets to 0 at the beginning of each interval.
unclassifiedMin	integer	**Ingress unclassified packets min value read**. This is the smallest value read by the counter during the collection interval. This value is only

CHAPTER 6

API Reference

acitoolkit package

Submodules

acibaseobject module

This module implements the Base Class for creating all of the ACI Objects.

class acitoolkit.acibaseobject.**BaseACIObject**(*name=None*, *parent=None*)
Bases: acitoolkit.aciSearch.AciSearch

This class defines functionality common to all ACI objects. Functions may be overwritten by inheriting classes.

Constructor initializes the basic object and should be called by the init routines of inheriting subclasses.

Parameters

- **name** – String containing the name of the object instance
- **parent** – Parent object within the acitoolkit object model.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

static check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

classmethod get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

classmethod **get_fault**(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

static **get_from_json**(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

static **get_table**(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

classmethod has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

static is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

classmethod mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

classmethod subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

classmethod unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acibaseobject.**BaseACIPhysModule**(*pod*, *node*, *slot*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

BaseACIPhysModule: base class for modules

Initialize the basic object. This should be called by the init routines of inheriting subclasses.

Parameters

- **pod** – pod id of module
- **node** – node id of module
- **slot** – slot id of module
- **parent** – optional parent object

classmethod get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

class acitoolkit.acibaseobject.**BaseACIPhysObject**(*name=''*, *parent=None*, *pod=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Base class for physical objects

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

classmethod check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

classmethod exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

classmethod get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_pod()
Gets pod_id :returns: id of pod

get_serial()
Gets serial number.

Returns serial number string

get_type()
Gets physical object type

Returns type string of the object.

static get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

class acitoolkit.acibaseobject.**BaseInterface**(*name=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Abstract class used to provide base functionality to other Interface classes.

Constructor initializes the basic object and should be called by the init routines of inheriting subclasses.

Parameters

- **name** – String containing the name of the object instance
- **parent** – Parent object within the acitoolkit object model.

get_port_channel_selector_json(*port_name*)
Get the JSON for the Port Channel selector

Parameters **port_name** – String containing the port name

Returns Dictonary containing the JSON for the Port Channel selector

get_port_selector_json()
Returns the port selector.

Returns

static is_dn_vpc(*dn*)
Check if the DN is a VPC

Parameters **dn** – String containing the DN

Returns True if the the DN is a VPC. False otherwise.

class acitoolkit.acibaseobject.**BaseRelation**(*item*, *status*, *relation_type=None*)
Bases: object

Class for all basic relations.

A relation consists of the following elements:

Parameters

- **item** – The object to which the relationship applies
- **status** – The status of the relationship. Valid values are 'attached' and 'detached'
- **relation_type** – Optional additional information to distinguish the relationship. Used in cases where more than 1 type of relation exists.

is_attached()

Returns True or False indicating whether the relation is attached. If a relation is detached, it will be deleted from the APIC when the configuration is pushed.

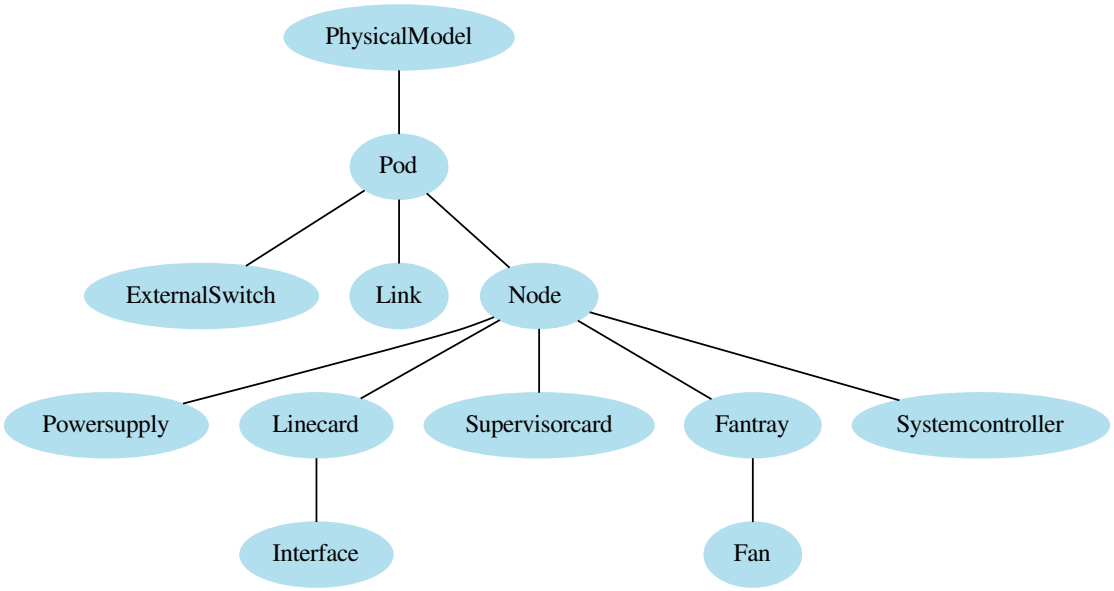
is_detached()

Returns True or False indicating whether the relation is detached. If a relation is detached, it will be deleted from the APIC when the configuration is pushed.

set_as_detached()
Sets the relation status to 'detached'

aciphysobject module

API Reference



ACI Toolkit module for physical objects

class acitoolkit.aciphysobject.**Cluster**(*name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Represents the global settings of the Cluster

Parameters **name** – String containing the name of this Cluster object.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the Clusters from the APIC.

Returns Instance of Cluster class.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_apics()

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_cluster_size()
reads information about the APIC cluster :return:

get_config_size()

Returns configured size of the cluster, i.e. # of APICs

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**ExternalSwitch**(*name=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

External Node. This class is for switch nodes that are connected to the pod, but are not ACI nodes, i.e. are not under control of the APIC. Examples would be external layer 2 switches, external routers, or hypervisor based switches.

This class will look as much as possible like the Node class recognizing that not as much information is available to the APIC about them as is available about ACI nodes. Nearly all of the information used to create this class comes from LLDP.

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the loose nodes from the APIC.

Parameters

- **session** – APIC session
- **parent** – optional parent object of type Topology

Returns list of ENodes

getRole()
retrieves the node role :returns: role

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*)
not yet fully implemented

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Gets serial number.

Returns serial number string

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

role
Getter for role. :return: role

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC

- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Fabric**(*session=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This is the root class for the acitoolkit. It is a container that can hold all of the other instances of the acitoolkit classes.

From this class, you can populate all of the children classes.

Initialization method that sets up the Fabric. :return:

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)

Parameters **session** –

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*session*, *include_concrete=False*)
Will return the entire tree of the fabric. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Fan**(*parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

Class for the fan of a fan tray

Initialize the basic fan.

Parameters **parent** – optional parent Fantray object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the fans from the APIC. If parent is specified, it will only get fantrays that are children of the the parent. The fantrays will also be added as children to the parent Node.

The fan object is derived mostly from the APIC 'eqptFan' class.

Parameters

- **session** – APIC session
- **parent** – optional parent fantray of class Fantray

Returns list of fans

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

static get_table(*modules*, *title=''*)
Will create table of fantry information :param title: :param modules:

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Fantray**(*pod*, *node*, *slot*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

Class for the fan tray of a node

Initialize the basic object. It will create the name of the fan tray and set the type before calling the base class __init__ method :param pod: pod id :param node: node id :param slot: slot id :param parent: optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the fantrays from the APIC. If parent is specified, it will only get fantrays that are children of the the parent. The fantrays will also be added as children to the parent Node.

The fantray object is derived mostly from the APIC 'eqptFt' class.

Parameters

- **session** – APIC session
- **parent** – optional parent switch of class Node

Returns list of fantrays

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

static get_table(*modules*, *title=''*)
Will create table of fantry information :param title: :param modules:

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Interface**(*interface_type*, *pod*, *node*, *module*, *port*, *parent=None*, *session=None*, *attributes=None*)
Bases: *acitoolkit.acibaseobject.BaseInterface*

This class defines a physical interface.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

classmethod create_from_name(*name*)

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

disable_cdp()
Disables CDP on this interface.

disable_lldp()
Disables LLDP on this interface.

enable_cdp()
Enables CDP on this interface.

enable_lldp()
Enables LLDP on this interface.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *pod_parent=None*, *node=None*, *module=None*, *port=None*)
Gets all of the physical interfaces from the APIC if no parent is specified. If a parent of type Linecard is specified, then only those interfaces on that linecard are returned and they are also added as children to that linecard.

If the pod, node, module and port are specified, then only that specific interface is read.

If the pod and node are specified, then only those interfaces are read

Parameters

- **session** – the instance of Session used for APIC communication
- **pod_parent** – Linecard instance to limit interfaces or pod number (optional)
- **node** – Node id string. This specifies the switch to read. (optional)
- **module** – Module id string. This specifies the module or slot of the port. (optional)
- **port** – Port number. This is the port to read. (optional)

Returns list of Interface instances

get_adjacent_port()
This will return the port ID of the port at the other end of the link.

For Access ports, it will only have a result if it is connected to a controller node.

If no link is found, then the result will be None. That does not mean that nothing is connected, just that a fabric link is not connected.

:returns : Port ID string

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –

- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Get the json for an interface. Returns a tuple since the json is required to be sent in multiple posts. A call to get_url will return the URLs which the JSON can be sent.

Returns Tuple containing the phys_domain, fabric, infra JSONs

get_parent()

Returns Parent of this object.

get_port_channel_selector_json(*port_name*)
Get the JSON for the Port Channel selector

Parameters **port_name** – String containing the port name

Returns Dictonary containing the JSON for the Port Channel selector

get_port_selector_json()
Returns the port selector.

Returns

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_serial()
getter for the serial number

Returns None

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
getter method for object.type

Returns the type

static get_url()
Gets URLs for physical domain, fabric, and infra.

Returns

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_cdp_disabled()
Returns whether this interface has CDP configured as disabled.

Returns True or False

is_cdp_enabled()
Returns whether this interface has CDP configured as enabled.

Returns True or False

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_dn_vpc(*dn*)
Check if the DN is a VPC

Parameters **dn** – String containing the DN

Returns True if the the DN is a VPC. False otherwise.

is_interface()
Returns whether this instance is considered an interface.

Returns True

is_lldp_disabled()
Returns whether this interface has LLDP configured as disabled.

Returns True or False

is_lldp_enabled()
Returns whether this interface has LLDP configured as enabled.

Returns True or False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

classmethod parse_dn(*dn*)
Parses the pod, node, module, port from a distinguished name of the interface.

Parameters **dn** – String containing the interface distinguished name

Returns interface_type, pod, node, module, port

static parse_name(*name*)
Parses a name that is of the form: <type> <pod>/<mod>/<port> :param name: Distinguished Name (dn)

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

push_to_apic(*session*)
Push the configuration to the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns Response class instance from the requests library. response.ok is True if request is sent successfully.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Linecard**(*arg0=None*, *arg1=None*, *slot=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

class for a linecard of a switch

Initialize the basic object. It will create the name of the linecard and set the type before calling the base class __init__ method. If arg1 is an instance of a Node, then pod, and node are derived from the Node and the slot_id is from arg0. If arg1 is not a Node, then arg0 is the pod, arg1 is the node id, and slot is the slot_id

In other words, this Linecard object can either be initialized by

>>> *lc = Linecard(slot_id, parent_switch)*

or

>>> *lc = Linecard(pod_id, node_id, slot_id)*

or

>>> *lc = Linecard(pod_id, node_id, slot_id, parent_switch)*

Parameters

- **arg0** – pod_id if arg1 is a node_id, slot_id if arg1 is of type Node
- **arg1** – node_id string or parent Node of type Node
- **slot** – slot_id if arg1 is node_id Not required if arg1 is a Node
- **parent** – parent switch of type Node. Not required if arg1 is used instead.

Returns None

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod **get**(*session*, *parent=None*)
Gets all of the linecards from the APIC. If parent is specified, it will only get linecards that are children of the the parent. The linecards will also be added as children to the parent Node.

The lincard object is derived mostly from the APIC 'eqptLC' class.

Parameters

- **session** – APIC session
- **parent** – optional parent of class Node

Returns list of linecards

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

static get_table(*linecards*, *super_title=''*)
Will create table of line card information :param super_title: :param linecards:

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Link**(*name=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

Link class, equivalent to the fabricLink object in APIC

Parameters **parent** – optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent_pod=None*, *node_id=None*)
Gets all of the Links from the APIC. If the parent_pod is specified, only links of that pod will be retrieved. If the parent_pod is a Pod object then the links will be added as children of that pod.

If node is specified, then only links of that originate at the specific node will be returned. If node is specified, pod must be specified.

Parameters

- **session** – APIC session
- **parent_pod** – Optional parent Pod object or identifier string.
- **node_id** – Optional node number string

Returns list of links

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_node1()
Returns the Node object that corresponds to the first node of the link. The Node must be a child of the Pod

that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Node object at first end of link

get_node2()
Returns the Node object that corresponds to the second node of the link. The Node must be a child of the Pod that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Node object at second end of link

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_port1()
Returns the Interface object that corresponds to the first port of the link. The port must be a child of the Linecard in the Node in the Pod that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Interface object at first end of link

get_port2()
Returns the Interface object that corresponds to the second port of the link. The port must be a child of the Linecard in the Node in the Pod that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Interface object at second end of link

get_port_id1()
Returns the port ID of the first end of the link in the format pod/node/slot/port

Returns port ID string

get_port_id2()
Returns the port ID of the second end of the link in the format pod/node/slot/port

Returns port ID string

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Gets serial number.

Returns serial number string

get_slot1()
Returns the Linecard object that corresponds to the first slot of the link. The Linecard must be a child of the Node in the Pod that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Linecard object at first end of link

get_slot2()

Returns the Linecard object that corresponds to the second slot of the link. The Linecard must be a child of

the Node in the Pod that this link is a member of, i.e. it must already have been read from the APIC. This can most easily be done by populating the entire physical heirarchy from the Pod down.

Returns Linecard object at second end of link

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Node**(*name=None*, *pod=None*, *node=None*, *role=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

Node : roughly equivalent to fabricNode

Parameters

- **pod** – String representation of the pod number
- **node** – String representation of the node number
- **name** – Name of the node
- **role** – Role of the node. Valid roles are None, 'spine', 'leaf', 'controller', 'loosenode'
- **parent** – Parent pod object of the node.

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*, *node_id=None*)
Gets all of the Nodes from the APIC. If the parent pod is specified, only nodes of that pod will be retrieved.

If parent pod and node_id is specified, only the matching switch will be retrieved.

APIC controller nodes will have a 'role' of 'controller', while switch nodes will have a 'role' of 'leaf' or 'spine'

Parameters

- **session** – APIC session
- **parent** – optional parent object or pod_id
- **node_id** – optional node_id of switch

Returns list of Nodes

getFabricSt()
retrieves the fabric state.

Returns fabric state

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_chassis_type()
Returns the chassis type of this node. The chassis type is derived from the model number. This is a chassis type that is compatible with Cisco's Cable Plan XML.

Returns chassis type of node of type str

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_firmware(*working_data*)
retrieves firmware version :param working_data:

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_health()
This will get the health of the switch node

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_role()
retrieves the node role :returns: role

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Gets serial number.

Returns serial number string

static get_table(*switches*, *title=''*)
Creates report of basic switch information :param switches: Array of Node objects :param title: optional title for this table

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

operSt
changed value to "oper_st" so this makes the class backward compatible :return:

populate_children(*deep=False*, *include_concrete=False*)
Will populate all of the children modules such as linecards, fantrays and powersupplies, of the node.

Parameters

- **deep** – boolean that when true will cause the entire sub-tree to be populated. When false, only the immediate children are populated
- **include_concrete** – boolean to indicate that concrete objects should also be populated

Returns List of children objects

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension='*', *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension='*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**PhysicalModel**(*session=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This is the root class for the physical part of the network. It's corrolary is the LogicalModel class. It is a container that can hold all of physical model instances. Initially this is only an instance of Pod.

From this class, you can populate all of the children classes.

Initialization method that sets up the Fabric. :return:

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session=None*, *parent=None*)
Method to get all of the PhysicalModels. It will get one and return it in a list. :param session: :param parent: :return: list of PhysicalModel

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Pod**(*pod*, *dn=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

Pod : roughly equivalent to fabricPod

Initialize the basic object. It will create the name of the pod and set the type before calling the base class __init__ method. Typically the pod_id will be 1.

Parameters

- **pod** – pod id string
- **dn** – distinguished name
- **parent** – optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the Pods from the APIC. Generally there will be only one.

Parameters

- **parent** – optional parent of class PhysicalModel
- **session** – APIC session

Returns list of Pods. Note that this will be a list even though there typically will only be one item in the list.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Gets serial number.

Returns serial number string

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Powersupply**(*pod*, *node*, *slot*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

class for a power supply in a node

Initialize the basic object. It will create the name of the powersupply and set the type before calling the base class __init__ method :param pod: pod id :param node: node id :param slot: slot id :param parent: optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the power supplies from the APIC. If parent is specified, it will only get power supplies that are children of the the parent. The power supplies will also be added as children to the parent Node.

The Powersupply object is derived mostly from the APIC 'eqptPsu' class.

Parameters

- **session** – APIC session
- **parent** – optional parent switch of class Node

Returns list of powersupplies

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

static get_table(*modules*, *super_title=''*)
Will create table of power supply information :param super_title: :param modules:

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Process**
Bases: *acitoolkit.acibaseobject.BaseACIPhysObject*

Class to hold information about a process running on a node - either switch or controller

Returns

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent*)

Parameters

- **session** –
- **parent** –

Returns

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Gets serial number.

Returns serial number string

static get_table(*aci_objects*, *title='Process'*)

Parameters

- **aci_objects** – list of process objects to build table for
- **title** – Title of the table

Returns Table

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Supervisorcard**(*pod*, *node*, *slot*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

Class representing the supervisor card of a switch

Initialize the basic object. This should be called by the init routines of inheriting subclasses.

Parameters

- **pod** – pod id
- **node** – node id
- **slot** – slot id
- **parent** – optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent_node=None*)
Gets all of the supervisor cards from the APIC. If parent is specified, it will only get the supervisor card that is a child of the the parent Node. The supervisor will also be added as a child to the parent Node.

The Supervisorcard object is derived mostly from the APIC 'eqptSupC' class.

If *parent_node* is a str, then it is the Node id of the switch for the supervisor.

Parameters

- **session** – APIC session
- **parent_node** – optional parent switch of class Node or the node id of a switch

Returns list of linecards

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

static get_table(*modules*, *super_title=''*)
Will create table of supervisor information :param super_title: :param modules:

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**Systemcontroller**(*pod*, *node*, *slot*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIPhysModule*

class of the motherboard of the APIC controller node

Initialize the basic object. It will create the name of the Systemcontroller and set the type before calling the base class __init__ method.

Parameters

- **pod** – pod id
- **node** – node id
- **slot** – slot id
- **parent** – optional parent object

add_child(*child_obj*)
Add a child to the children list. All children must be unique so it will first delete the child if it already exists.

Parameters **child_obj** – a child object to be added as a child to this object. This will be put into the _children list.

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_parent(*parent*)
If a parent is specified, it will check that it is the correct class of parent If not, then an exception is raised. :param parent: :return:

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

exists(*session*, *phys_obj*)
Check if an apic phys_obj exists on the APIC. Returns True if the phys_obj does exist.

Parameters

- **session** – APIC session to use when accessing the APIC controller.
- **phys_obj** – The object that you are checking for.

Returns True if the phys_obj exists, False if it does not.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod **get**(*session*, *parent=None*)
Gets all of the System controllers from the APIC. This information comes from the APIC 'eqptBoard' class.

If parent is specified, it will only get system controllers that are children of the the parent. The system controlles will also be added as children to the parent Node.

Parameters

- **session** – APIC session
- **parent** – parent Node

Returns list of Systemcontrollers

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*child_type=None*)
Returns the list of children. If childType is provided, then it will return all of the children of the matching type.

Parameters **child_type** – This optional parameter will cause this method to return only those children that match the type of childType. If this parameter is ommitted, then all of the children will be returned.

Returns list of children

get_deep(*session*, *include_concrete=False*)
Will return the atk object and the entire tree under it. :param session: APIC session to use :param include_concrete: flag to indicate that concrete objects should also be included :return:

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the object

Returns JSON of contained Interfaces

get_name()
Gets name.

Returns Name string

get_node()
Gets node id

Returns id of node

get_obj(*session*, *apic_classes*, *parent_node*)
Gets all of the Nodes from the APIC. This is called by the module specific get() methods. The parameters passed include the APIC object class, apic_classes, so that this will work for different kinds of modules.

Parameters

- **parent_node** – parent object or node id
- **session** – APIC session to use when retrieving the nodes
- **apic_classes** – The object class in APIC to retrieve

Returns list of module objects derived from the specified apic_classes

get_parent()

Returns Parent of this object.

get_pod()
Gets pod_id :returns: id of pod

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_serial()
Returns the serial number. :returns: serial number string

get_slot()
Gets slot id

Returns slot id

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_type()
Gets physical object type

Returns type string of the object.

get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no fmt parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the fmt string appended.

Parameters **fmt** – optional fmt string

Returns Nothing - physical objects are not modifiable

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.aciphysobject.**WorkingData**(*session=None*, *toolkit_class=None*, *url=None*, *deep=False*, *include_concrete=False*)

Bases: object

This class will hold the entire json tree from topSystem down, for a switch. The attributes of a specific class can be retrieved in which case it will be as a list of objects. It will allow all children of an object to be retrieved result is list of objects It will allow an instance of a class to be retrieved returned as a single object.

add(*session=None*, *toolkit_class=None*, *url=None*, *deep=False*, *include_concrete=False*)

Parameters

- **session** –
- **toolkit_class** –
- **url** –
- **deep** –
- **include_concrete** –

Returns

build_vnid_dictionary()

Will build a dictionary that is indexed by vnid and will return context or bridge_domain and the name of that segment. :param self:

get_class(*class_name*)

returns all the objects of a given class :param class_name: The name of the class you are looking for.

get_object(*dname*)

Will return the object specified by dn. :param dname: Distinguished Name (dn)

get_subtree(*class_name*, *dname*)

will return list of matching classes and their attributes

It will get all classes that are classes under dn. :param class_name: name of class you are looking for :param dname: Distinguished Name (dn)

Future Work

Future work items to be added to aciphysobject include:

- Add an events sub-object. This would work similarly to the stats object.
- Add a top level object called Topology that would have Pod as its child. This object would also contain devices attached to the fabric, so called loose nodes, external links, and discovered end-points.

acisession module

This module contains the Session class that controls communication with the APIC.

class acitoolkit.acisession.**Session**(*url*, *uid*, *pwd=None*, *cert_name=None*, *key=None*, *verify_ssl=False*, *appcenter_user=False*, *subscription_enabled=True*, *proxies=None*)

Bases: object

Session class This class is responsible for all communication with the APIC.

Parameters

- **url** – String containing the APIC URL such as `https://1.2.3.4`
- **uid** – String containing the username that will be used as part of the the APIC login credentials.
- **pwd** – String containing the password that will be used as part of the the APIC login credentials.
- **cert_name** – String containing the certificate name that will be used as part of the the APIC certificate authentication credentials.
- **key** – String containing the private key file name that will be used as part of the the APIC certificate authentication credentials.
- **verify_ssl** – Used only for SSL connections with the APIC. Indicates whether SSL certificates must be verified. Possible values are True and False with the default being False.
- **appcenter_user** – Set True when using certificate authentication from the context of an APIC appcenter app
- **proxies** – Optional dictionary containing the proxies passed directly to the Requests library

close()
Close the session

deregister_login_callback(*callback_fn*)
Delete the registration of a callback function that was registered via the register_login_callback function.

Parameters **callback_fn** – function to be deregistered

get(*url*, *timeout=None*)
Perform a REST GET call to the APIC.

Parameters **url** – String containing the URL that will be used to send the object data to the APIC.

Returns Response class instance from the requests library. response.ok is True if request is sent successfully. response.json() will return the JSON data sent back by the APIC.

get_event(*url*)
Get an event for a particular URL. Used internally by the class and instance subscriptions.

Parameters **url** – URL string belonging to subscription

Returns Object belonging to the instance or class that the subscription was made.

get_event_count(*url*)
Check the number of subscription events for a particular APIC URL

Parameters **url** – URL string belonging to subscription

Returns Interger number of events in event queue

has_events(*url*)
Check if there are events for a particular URL. Used internally by the class and instance subscriptions.

Parameters **url** – URL string belonging to subscription

Returns True or False. True if an event exists for this subscription.

invoke_login_callbacks()
Invoke registered callback functions when the session performs a successful relogin attempt after disconnecting from the APIC.

is_subscribed(*url*)
Check if subscribed to events for a particular URL.

Parameters **url** – URL string to issue subscription

logged_in()
Returns whether the session is logged in to the APIC

Returns True or False. True if the session is logged in to the APIC.

login(*timeout=None*)
Initiate login to the APIC. Opens a communication session with the APIC using the python requests library.

Returns Response class instance from the requests library. response.ok is True if login is successful.

push_to_apic(*url*, *data*, *timeout=None*)
Push the object data to the APIC

Parameters

- **url** – String containing the URL that will be used to send the object data to the APIC.
- **data** – Dictionary containing the JSON objects to be sent to the APIC.

Returns Response class instance from the requests library. response.ok is True if request is sent successfully.

refresh_login(*timeout=None*)
Refresh the login to the APIC

Parameters **timeout** – Integer containing the number of seconds for connection timeout

Returns Instance of requests.Response

register_login_callback(*callback_fn*)
Register a callback function that will be called when the session performs a successful relogin attempt after disconnecting from the APIC.

Parameters **callback_fn** – function to be called

resubscribe()
Resubscribe to the current subscriptions. Used by the login thread after a re-login

Returns None

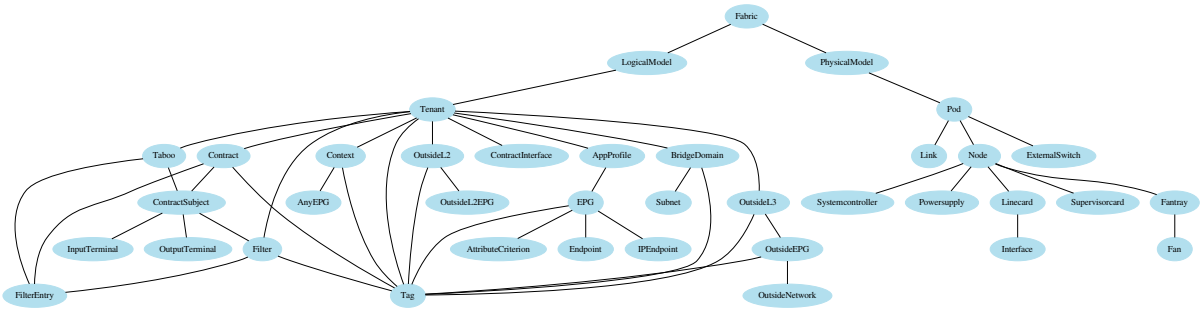
subscribe(*url*, *only_new=False*)
Subscribe to events for a particular URL. Used internally by the class and instance subscriptions.

Parameters **url** – URL string to issue subscription

unsubscribe(*url*)
Unsubscribe from events for a particular URL. Used internally by the class and instance subscriptions.

Parameters **url** – URL string to remove issue subscription

acitoolkit module



Main ACI Toolkit module This is the main module that comprises the ACI Toolkit.

class acitoolkit.acitoolkit.**AnyEPG**(*epg_name*, *parent=None*)

Bases: *acitoolkit.acitoolkit.CommonEPG*

AnyEPG class, roughly equivalent to vz:Any

Parameters

- **epg_name** – String containing the name of this EPG
- **parent** – Instance of the AppProfile class representing the Application Profile where this EPG is contained.

add_child(*obj*)

Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)

Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)

Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)

This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

consume(*contract*)

Make this EPG consume a Contract

Parameters **contract** – Contract class instance to be consumed by this EPG.

Returns True

consume_cif(*contract_interface*)

Make this EPG consume a ContractInterface

Parameters **contract_interface** – ContractInterface class instance to be consumed by this EPG.

Returns True

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_consume(*contract*)
Check if this EPG consumes a specific Contract

Parameters **contract** – Instance of Contract class to check if it is consumed by this EPG.

Returns True or False. True if the EPG does consume the Contract.

does_consume_cif(*contract_interface*)
Check if this EPG consumes a specific Contract

Parameters **contract_interface** –

Returns True or False. True if the EPG does consume the ContractInterface.

does_protect(*taboo*)
Check if this EPG is protected by a specific Taboo.

Parameters **taboo** – Instance of Taboo class to check if it protects this EPG.

Returns True or False. True if the EPG is protected by the Taboo.

does_provide(*contract*)
Check if this EPG provides a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is provided by this EPG.

Returns True or False. True if the EPG does provide the Contract.

dont_consume(*contract*)
Make this EPG not consume a Contract. It does not check to see if the Contract was already consumed

Parameters **contract** – Instance of Contract class to be no longer consumed by this EPG.

Returns True

dont_consume_cif(*contract_interface*)
Make this EPG not consume a ContractInterface. It does not check to see if the ContractInterface was already consumed

Parameters **contract_interface** –

Returns True

dont_protect(*taboo*)
Make this EPG not protected by a Taboo

Parameters **taboo** – Instance of Taboo class to no longer protect this EPG.

Returns True

dont_provide(*contract*)
Make this EPG not provide a Contract

Parameters **contract** – Instance of Contract class to be no longer provided by this EPG.

Returns True

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *parent=None*, *tenant=None*)
Gets all of the EPGs from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Instance of the AppProfile class used to limit the EPGs retreived from the APIC.
- **tenant** – Instance of Tenant class used to limit the EPGs retreived from the APIC.

Returns List of CommonEPG instances (or EPG instances if called from EPG class)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consumed(*deleted=False*)
Get all of the Contracts consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are consumed or that the consumed was marked as deleted

Returns List of Contract objects that are consumed by the EPG.

get_all_consumed_cif(*deleted=False*)
Get all of the ContractInterfaces consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get ContractInterfaces that are consumed or that the consumed was marked as deleted

Returns List of ContractInterface objects that are consumed by the EPG.

get_all_protected(*deleted=False*)
Get all of the Taboos protecting this EPG

Parameters **deleted** – Boolean indicating whether to get Taboos that are protected or that the protected was marked as deleted

Returns List of Taboo objects that are protecting the EPG.

get_all_provided(*deleted=False*)
Get all of the Contracts provided by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are provided or that the provided was marked as deleted

Returns List of Contract objects that are provided by the EPG.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interfaces that this EPG is attached. The default is to get list of 'attached' interfaces. If 'status' is set to 'detached' it will return the list of detached Interface objects (Those EPGs which are no longer attached to an Interface, but the configuration is not yet pushed to the APIC.)

Parameters **status** – 'attached' or 'detached'. Defaults to 'attached'.

Returns List of Interface objects

get_json()
Returns json representation of the EPG

Returns json dictionary of the EPG

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension='*')
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

protect(*taboo*)
Make this EPG protected by a Taboo

Parameters **taboo** – Instance of Taboo class to protect this EPG.

Returns True

provide(*contract*)
Make this EPG provide a Contract

Parameters **contract** – Instance of Contract class to be provided by this EPG.

Returns True

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**AppProfile**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

The AppProfile class is used to represent the Application Profiles within the acitoolkit object model. In the APIC model, this class is roughly equivalent to the fvAp class.

Parameters

- **name** – String containing the Application Profile name
- **parent** – An instance of Tenant class representing the Tenant which contains this Application Profile.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *tenant*)
Gets all of the Application Profiles from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the Application Profiles retreived from the APIC

Returns List of AppProfile objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the AppProfile object.

Returns json dictionary of fvAp

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*app_profiles*, *title=''*)
Will create table of app_profile information for a given tenant

Parameters

- **title** –
- **app_profiles** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**AttributeCriterion**(*name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

AttributeCriterion : roughly equivalent to fvCrtrn

Initializes the AttributeCriterion with a name and optionally an EPG parent

Parameters

- **name** – String containing the AttributeCriterion name

- **parent** – Instance of the EPG class representing where this AttributeCriterion is defined

Returns Instance of AttributeCriterion class

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_ip_address(*ip_addr*)
Add an IP address as an attribute

Parameters **ip_addr** – String containing the IP address

Returns None

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_ip_addresses()
return the list of IP addresses

get_json()
Returns JSON representation of the AttributeCriterion :return:

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

match
Return the match property :return: String containing the match property. Possible values are 'any' or 'all'

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**BGPSession**(*name*, *router_id=None*, *peer_ip=None*, *node_id=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Creates an BGP router interface that can be attached to a L3 interface. This interface defines the BGP AS, authentication, etc.

Parameters

- **name** – String containing the name of this BGPSession object.
- **router_id** – String containing the IPv4 router-id
- **peer_ip** – String containing the IP address of the BGP peer Default is None.
- **node_id** – String Containing the node-id (e.g. '101')

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of BGPSession

Returns json dictionary of BGP Session

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

static is_bgp()

Returns True if this interface is an BGP interface. In the case of BGPSession instances, this is always True.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Returns whether this instance is considered an interface.

Returns True

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**BaseContract**(*contract_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

BaseContract : Base class for Contracts and Taboos

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_filter_entries(*direction='bidirectional-only'*)
Get all of the filter entries contained within this Contract/Taboo

Parameters **direction** – String containing the type of filter entries to gather Valid values are 'bidirectional-only', 'input-only', 'output-only', 'all' Default is 'bidirectional-only'

Returns List of FilterEntry instances

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the contract

Returns json dictionary of the contract

get_parent()

Returns Parent of this object.

get_scope()
Get the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

classmethod **mask_class_from_graphs**()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

Parameters **scope** – String containing one of the following 'context', 'global', 'tenant', or 'application-profile'

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**BaseMonitorClass**
Bases: object

Base class for monitoring policies. These are methods that can be used on all monitoring objects.

add_collection_policy(*coll_obj*)
Add a collection policy.

Parameters **coll_obj** – A collection policy object of type CollectionPolicy

add_stats(*stat_obj*)
Adds a stats family object.

Parameters **stat_obj** – Statistics family object of type MonitorStats.

add_target(*target_obj*)
Add a target object.

Parameters **target_obj** – target object of type MonitorTarget

get_parent()

Returns parent object

isModified()
Returns True if this policy and any children have been modified or created and not been written to the APIC

remove_collection_policy(*collection*)
Remove a collection_policy object. The object to remove is identified by its granularity, e.g. '5min', '15min', etc. This string can be found in the 'CollectionPolicy.granularity' attribute of the object.

Parameters **collection** – CollectionPolicy to remove.

remove_stats(*stats_family*)
Remove a stats family object. The object to remove is identified by a string, e.g. 'ingrPkts', or 'egrTotal'. This string can be found in the 'MonitorStats.scope' attribute of the object.

Parameters **stats_family** – Statistics family string.

remove_target(*target*)
Remove a target object. The object to remove is identified by a string, e.g 'l1PhysIf'. This string can be found in the 'MonitorTarget.scope' attribute of the object.

Parameters **target** – target to remove.

set_description(*description*)

Sets the description of the MonitorStats.

Parameters **description** – String to use as the description

set_name(*name*)

Sets the name of the MonitorStats.

Parameters **name** – String to use as the name

class acitoolkit.acitoolkit.**BaseSubnet**(*name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Base class for Subnet and OutsideNetwork

Parameters

- **name** – String containing the name of this instance.
- **parent** – An instance of the parent class.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

addr
Subnet address

Returns String containing the subnet default gateway IP address and mask e.g. "1.2.3.4/24"

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication

- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_addr()
Get the subnet address

Returns The subnet address as a string in the form of <ipaddr>/<mask>

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_scope()
Get the subnet scope

Returns The subnet scope as a string

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

ip
IP address of the subnet in the form of Address/mask e.g. 10.1.1.1/16

Returns String containing the IP address

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_addr(*addr*)
Set the subnet address

Parameters **addr** – The subnet default gateway address as a string in the form of <ipaddr>/<mask>

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the subnet scope

Parameters **scope** – String containing the subnet scope

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**BaseTerminal**(*terminal_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Base class for Input terminal and output terminal

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_filter(*filter_obj*)
Add Filter to the Terminal, roughly equivalent to vzRsFiltAtt

Parameters **filter_obj** – Instance of Filter class. Represents a Filter that is added to the Terminal. Multiple Filters can be assigned to a single Terminal.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_filters()
Get all of the filters that are attached to this Terminal.

Returns List of Filter objects

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the Terminal Object

Returns json dictionary of the ContractSubject

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

classmethod **mask_class_from_graphs**()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**BridgeDomain**(*bd_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

BridgeDomain : roughly equivalent to fvBD

Parameters

- **bd_name** – String containing the name of this BridgeDomain object.
- **parent** – An instance of Tenant class representing the Tenant which contains this BridgeDomain.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_context(*context*)
Set the Context for this BD

Parameters **context** – Context to assign this BridgeDomain

add_l3out(*l3out*)
Set the L3Out for this BD

Parameters **l3out** – OutsideL3 to assign this BridgeDomain

add_subnet(*subnet*)
Add a subnet to this BD.

Parameters **subnet** – Instance of Subnet class to add to this BridgeDomain.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *tenant*)
Gets all of the Bridge Domains from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the BridgeDomain instances retreived from the APIC

Returns List of BridgeDomain objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_arp_flood()
Get the ARP flooding policy for this BD

Returns a string containing the ARP flooding policy of the BridgeDomain

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_context()
Get the Context for this BD

Returns Instance of Context class that this BridgeDomain is assigned.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

static get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the bridge domain

Returns json dictionary of bridge domain

get_l3out()

Returns List of OutsideL3 objects

get_mac()
Get the mac address for the BD

Returns string containing the mac address of the BD (e.g. 00:22:BD:F8:19:FF)

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_subnets()
Get all of the subnets on this BD.

Returns List of Subnet instances assigned to this BridgeDomain.

static get_table(*bridge_domains*, *title=''*)
Will create table of context information

Parameters

- **title** –
- **bridge_domains** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

get_unicast_route()
Get the Unicast Routing policy for this BD

Returns a string containing the unicast routing policy of the BridgeDomain

get_unknown_mac_unicast()
Gets the unknown mac unicast for this BD

Returns unknown mac unicast of the BridgeDomain

get_unknown_multicast()
Gets the unknown multicast for this BD

Returns unknown multicast of the BridgeDomain

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_context()
Check if the Context has been set for this BD

Returns True or False. True if this BridgeDomain is assigned to a Context.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension='*')
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_l3out()
Check if this BD has an OutsideL3 :return: True if the BD has an OutsideL3 configured. False, otherwise.

has_parent()
returns True if this object has a parent

Returns bool

has_subnet(*subnet*)
Check if the BD has this particular subnet.

Parameters **subnet** –

Returns True or False. True if this BridgeDomain has this particular Subnet.

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_arp_flood()
Check if ARP flooding is enabled

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

is_unicast_route()
Check if unicast routing is enabled

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_context()
Remove the assigned Context from this BD

remove_subnet(*subnet*)
Remove a subnet from this BD

Parameters **subnet** – Instance of Subnet class to remove from this BridgeDomain.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_arp_flood(*arp_value*)
Set the ARP flood for this BD

Parameters **arp_value** – arp to assign this BridgeDomain

set_mac(*mac*)
Set the mac address for the BD

Parameters **mac** – string mac address (XX:XX:XX:XX:XX:XX)

set_multidestination(*multidestination*)
Set the multidestination flood policy for this BD

Parameters **multidestination** – policy to assign this BridgeDomain

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_unicast_route(*route*)
Set the unicast route for this BD

Parameters **route** – route to assign this BridgeDomain

set_unknown_mac_unicast(*unicast*)
Set the unknown mac unicast for this BD

Parameters **unicast** – Unicast to assign this BridgeDomain

set_unknown_multicast(*multicast*)
Set the unknown multicast for this BD

Parameters **multicast** – Multicast to assign this BridgeDomain

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**CollectionPolicy**(*parent*, *granularity*, *retention*, *adminState='enabled'*)

Bases: *acitoolkit.acitoolkit.BaseMonitorClass*

This class is a child of a MonitorPolicy object, MonitorTarget object or a MonitorStats object. It is where the statistics collection policy is actually specified. It applies to all of the statistics that are at the scope level of the parent object, i.e. all, specific to a target, or specific to a statistics family. What is specified in the CollectionPolicy is the time granularity of the collection and how much history to retain. For example, the granularity might be 5 minutes (5min) or 1 hour (1h). How much history to retain is similarly specified. For example you might specify that it be kept for 10 days (10d) or 2 years (2year).

If the CollectionPolicy is a child of a MonitorStats object, it can optionally have children that specify the policy for raising threshold alarms on the fields in the stats family specified in the MonitorStats object. This has yet to be implemented.

This object is roughly the same as the statsColl and statsHierColl objects in the APIC.

The CollectionPolicy must always be initialized with a parent object of type MonitorPolicy, MonitorTarget or MonitorStats. The granularity must also be specifically specified. The retention period can be specified, set to "none", or set to "inherited". Note that the "none" value is a string, not the Python None. When the retention period is set to "none" there will be no historical stats kept. However, assuming collection is enabled, stats will be kept for the current time period.

If the retention period is set to "inherited", the value will be inherited from the less specific policy directly above this one. The same applies to the adminState value. It can be 'disabled', 'enabled', or 'inherited'. If 'disabled', the current scope of counters are not gathered. If enabled, they are gathered. If 'inherited', it will be according to the next higher scope.

Having the 'inherited' option on the retention and administrative status allows these items independently controlled at the current stats granularity. For example, you can specify that ingress unknown packets are gathered every 15 minutes by setting adding a collection policy that specifies a 15 minutes granularity and an adminState of 'enabled' under a MonitorStats object that sets the scope to be ingress unknown packets. This might override a higher level policy that disabled collection at a 15 minute interval. However, you can set the retention in that same object to be "inherited" so that this specific policy does not change the retention behavior from that of the higher, less specific, policy.

When the CollectionPolicy is a child at the top level, i.e. of the MonitorPolicy, the 'inherited' option is not allowed because there is no higher level policy to inherit from. If this were to happen, 'inherited' will be treated as 'enabled'.

Parameters

- **parent** – Parent object that this collection policy should be applied to. This must be an object of type MonitorStats, MonitorTarget, or MonitorPolicy.
- **granularity** – String specifying the time collection interval or granularity of this policy. Possible values are: ['5min', '15min', '1h', '1d', '1w', '1mo', '1qtr', '1year'].

- **retention** – String specifying how much history to retain the collected statistics for. The retention will be for time units of the granularity specified. Possible values are ['none', 'inherited', '5min', '15min', '1h', '1d', '1w', '10d', '1mo', '1qtr', '1year', '2year', '3year'].
- **adminState** – Administrative status. String to specify whether stats should be collected at the specified granularity. Possible values are ['enabled', 'disabled', 'inherited']. The default if not specified is 'enabled'.

add_collection_policy(*coll_obj*)
Add a collection policy.

Parameters **coll_obj** – A collection policy object of type CollectionPolicy

add_stats(*stat_obj*)
Adds a stats family object.

Parameters **stat_obj** – Statistics family object of type MonitorStats.

add_target(*target_obj*)
Add a target object.

Parameters **target_obj** – target object of type MonitorTarget

get_parent()

Returns parent object

granularityEnum = ['5min', '15min', '1h', '1d', '1w', '1mo', '1qtr', '1year']

isModified()
Returns True if this policy and any children have been modified or created and not been written to the APIC

remove_collection_policy(*collection*)
Remove a collection_policy object. The object to remove is identified by its granularity, e.g. '5min', '15min', etc. This string can be found in the 'CollectionPolicy.granularity' attribute of the object.

Parameters **collection** – CollectionPolicy to remove.

remove_stats(*stats_family*)
Remove a stats family object. The object to remove is identified by a string, e.g. 'ingrPkts', or 'egrTotal'. This string can be found in the 'MonitorStats.scope' attribute of the object.

Parameters **stats_family** – Statistics family string.

remove_target(*target*)
Remove a target object. The object to remove is identified by a string, e.g 'l1PhysIf'. This string can be found in the 'MonitorTarget.scope' attribute of the object.

Parameters **target** – target to remove.

retentionEnum = ['none', 'inherited', '5min', '15min', '1h', '1d', '1w', '10d', '1mo', '1qtr', '1year', '2year', '3year']

setAdminState(*adminState*)
Sets the administrative status.

Parameters **adminState** – Administrative status. String to specify whether stats should be collected at the specified granularity. Possible values are ['enabled', 'disabled', 'inherited']. The default if not specified is 'enabled'.

setRetention(*retention*)

Sets the retention period.

Parameters **retention** – String specifying how much history to retain the collected statistics for. The retention will be for time units of the granularity specified. Possible values are ['none', 'inherited', '5min', '15min', '1h', '1d', '1w', '10d', '1mo', '1qtr', '1year', '2year', '3year'].

set_description(*description*)

Sets the description of the MonitorStats.

Parameters **description** – String to use as the description

set_name(*name*)

Sets the name of the MonitorStats.

Parameters **name** – String to use as the name

class acitoolkit.acitoolkit.**CommonEPG**(*epg_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Base class for EPG and OutsideEPG. Not meant to be instantiated directly

Parameters

- **epg_name** – String containing the name of this EPG
- **parent** – Instance of the AppProfile class representing the Application Profile where this EPG is contained.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

consume(*contract*)
Make this EPG consume a Contract

Parameters **contract** – Contract class instance to be consumed by this EPG.

Returns True

consume_cif(*contract_interface*)
Make this EPG consume a ContractInterface

Parameters **contract_interface** – ContractInterface class instance to be consumed by this EPG.

Returns True

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_consume(*contract*)
Check if this EPG consumes a specific Contract

Parameters **contract** – Instance of Contract class to check if it is consumed by this EPG.

Returns True or False. True if the EPG does consume the Contract.

does_consume_cif(*contract_interface*)
Check if this EPG consumes a specific Contract

Parameters **contract_interface** –

Returns True or False. True if the EPG does consume the ContractInterface.

does_protect(*taboo*)
Check if this EPG is protected by a specific Taboo.

Parameters **taboo** – Instance of Taboo class to check if it protects this EPG.

Returns True or False. True if the EPG is protected by the Taboo.

does_provide(*contract*)
Check if this EPG provides a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is provided by this EPG.

Returns True or False. True if the EPG does provide the Contract.

dont_consume(*contract*)
Make this EPG not consume a Contract. It does not check to see if the Contract was already consumed

Parameters **contract** – Instance of Contract class to be no longer consumed by this EPG.

Returns True

dont_consume_cif(*contract_interface*)
Make this EPG not consume a ContractInterface. It does not check to see if the ContractInterface was already consumed

Parameters **contract_interface** –

Returns True

dont_protect(*taboo*)
Make this EPG not protected by a Taboo

Parameters **taboo** – Instance of Taboo class to no longer protect this EPG.

Returns True

dont_provide(*contract*)
Make this EPG not provide a Contract

Parameters **contract** – Instance of Contract class to be no longer provided by this EPG.

Returns True

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*, *tenant=None*)
Gets all of the EPGs from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Instance of the AppProfile class used to limit the EPGs retreived from the APIC.
- **tenant** – Instance of Tenant class used to limit the EPGs retreived from the APIC.

Returns List of CommonEPG instances (or EPG instances if called from EPG class)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consumed(*deleted=False*)
Get all of the Contracts consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are consumed or that the consumed was marked as deleted

Returns List of Contract objects that are consumed by the EPG.

get_all_consumed_cif(*deleted=False*)
Get all of the ContractInterfaces consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get ContractInterfaces that are consumed or that the consumed was marked as deleted

Returns List of ContractInterface objects that are consumed by the EPG.

get_all_protected(*deleted=False*)
Get all of the Taboos protecting this EPG

Parameters **deleted** – Boolean indicating whether to get Taboos that are protected or that the protected was marked as deleted

Returns List of Taboo objects that are protecting the EPG.

get_all_provided(*deleted=False*)
Get all of the Contracts provided by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are provided or that the provided was marked as deleted

Returns List of Contract objects that are provided by the EPG.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interfaces that this EPG is attached. The default is to get list of 'attached' interfaces. If 'status' is set to 'detached' it will return the list of detached Interface objects (Those EPGs which are no longer attached to an Interface, but the configuration is not yet pushed to the APIC.)

Parameters **status** – 'attached' or 'detached'. Defaults to 'attached'.

Returns List of Interface objects

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

protect(*taboo*)
Make this EPG protected by a Taboo

Parameters **taboo** – Instance of Taboo class to protect this EPG.

Returns True

provide(*contract*)
Make this EPG provide a Contract

Parameters **contract** – Instance of Contract class to be provided by this EPG.

Returns True

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Context**(*context_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Context : roughly equivalent to fvCtx

Parameters

- **context_name** – String containing the Context name
- **parent** – An instance of Tenant class representing the Tenant which contains this Context.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *tenant=None*)
Gets all of the Contexts from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the Contexts retreived from the APIC

Returns List of Context objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_allow_all()
Returns the allow_all value from this Context. When set, contracts will not be enforced in this context.

Returns True or False.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of fvCtx object

Returns json dictionary of fvCtx object

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*contexts*, *title=''*)
Will create table of context information

Parameters

- **title** –
- **contexts** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False

- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_allow_all(*value=True*)
Set the allow_all value. When set, contracts will not be enforced in this context.

Parameters **value** – True or False. Default is True.

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Contract**(*contract_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.BaseContract*

Contract : Class for Contracts

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod **get**(*session*, *tenant*)
Gets all of the Contracts from the APIC for a particular tenant.

Parameters

- **tenant** –
- **session** –

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consuming_epgs(*deleted=False*)
Get all of the EPGs consuming this contract

Parameters **deleted** – Boolean indicating whether to get EPGs that are consuming or that the consuming relationship was marked as deleted

Returns List of EPG instances

get_all_filter_entries(*direction='bidirectional-only'*)
Get all of the filter entries contained within this Contract/Taboo

Parameters **direction** – String containing the type of filter entries to gather Valid values are 'bidirectional-only', 'input-only', 'output-only', 'all' Default is 'bidirectional-only'

Returns List of FilterEntry instances

get_all_providing_epgs(*deleted=False*)
Get all of the EPGs providing this contract

Parameters **deleted** – Boolean indicating whether to get EPGs that are providing or that the providing relationship was marked as deleted

Returns List of EPG instances

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the contract

Returns json dictionary of the contract

get_parent()

Returns Parent of this object.

get_scope()
Get the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*contracts*, *title=''*)
Will create of each contract

Parameters

- **title** –
- **contracts** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

Parameters **scope** – String containing one of the following 'context', 'global', 'tenant', or 'application-profile'

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**ContractInterface**(*contractif_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

ContractInterface : roughly equivalent to vzCPIf

Parameters

- **contractif_name** – String containing the ContractInterface name
- **parent** – An instance of Tenant class representing the Tenant which contains this ContractInterface.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_import_contract(*contract*)
Check if this ContractInterface imports a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is imported by this ContractInterface.

Returns True or False. True if the ContractInterface does import the Contract.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *tenant=None*)
Gets all of the ContractInterfaces from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the ContractInterfaces retrieved from the APIC

Returns List of ContractInterface objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_import_contract(*deleted=False*)
Get the specific Contract that this ContractInterface is importing.

Parameters **contract** – Instance of Contract class that is imported by this ContractInterface.

Returns Contract class instance or None if not importing a contract

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of vzCPIf object

Returns json dictionary of vzCPIf object

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_import_contract()
Check if the ContractInterface has any imported Contract :return: True or False. True if the ContractInterface does import a Contract.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

import_contract(*contract*)
Set the Contract that is imported by this ContractInterface

Parameters **contract** – Instance of Contract

Returns None

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**ContractSubject**(*subject_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

ContractSubject : roughly equivalent to vzSubj

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_filter(*filter_obj*)
Add Filter to the ContractSubject, roughly equivalent to vzRsSubjFiltAtt

Parameters **filter_obj** – Instance of Filter class. Represents a Filter that is added to the ContractSubject. Multiple Filter can be assigned to a single ContractSubject.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_filters(*deleted=False*)
Get all of the filters that are attached to this ContractSubject.

Returns List of Filter objects

static get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the ContractSubject or TabooContractSubject

Returns json dictionary of the ContractSubject or TabooContractSubject

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**EPG**(*epg_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.CommonEPG*

EPG : roughly equivalent to fvAEPg

Initializes the EPG with a name and, optionally, an AppProfile parent. If the parent is specified and is not an AppProfile, an error will occur.

Parameters

- **epg_name** – String containing the name of the EPG.
- **parent** – Instance of the AppProfile class representing the Application Profile where this EPG is contained.

add_bd(*bridgedomain*)
Add BridgeDomain to the EPG, roughly equivalent to fvRsBd

Parameters **bridgedomain** – Instance of BridgeDomain class. Represents the BridgeDomain that this EPG will be assigned. An EPG can only be assigned to a single BridgeDomain.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_infradomain(*infradomain*)
Add Infrastructure Domain to the EPG

Parameters **infradomain** – Instance of InfraDomain class.

add_static_leaf_binding(*leaf_id*, *encap_type*, *encap_id*, *encap_mode='regular'*, *immediacy='lazy'*, *pod=1*)
Adds a static leaf binding to this EPG.

Parameters

- **leaf_id** – Integer containing the node ID (e.g. 101)
- **encap_type** – String containing the encapsulation type. Valid values are 'vlan', 'vxlan', or 'nvgre'.
- **encap_id** – String containing the encapsulation specific identifier representing the virtual L2 network (i.e. for VXLAN, this is the numeric value of the VNID).
- **encap_mode** – String containing the encapsulation mode. Use

"regular" for normal dot1q tagged traffic, "untagged" for traffic reaching the leaf without any dot1q tags, and "native" for traffic tagged with a 802.1P tag.

Parameters

- **immediacy** – String containing either "immediate" or "lazy"
- **pod** – Integer containing the ACI Pod where the supplied leaf is located.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

consume(*contract*)
Make this EPG consume a Contract

Parameters **contract** – Contract class instance to be consumed by this EPG.

Returns True

consume_cif(*contract_interface*)
Make this EPG consume a ContractInterface

Parameters **contract_interface** – ContractInterface class instance to be consumed by this EPG.

Returns True

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_consume(*contract*)
Check if this EPG consumes a specific Contract

Parameters **contract** – Instance of Contract class to check if it is consumed by this EPG.

Returns True or False. True if the EPG does consume the Contract.

does_consume_cif(*contract_interface*)
Check if this EPG consumes a specific Contract

Parameters **contract_interface** –

Returns True or False. True if the EPG does consume the ContractInterface.

does_protect(*taboo*)
Check if this EPG is protected by a specific Taboo.

Parameters **taboo** – Instance of Taboo class to check if it protects this EPG.

Returns True or False. True if the EPG is protected by the Taboo.

does_provide(*contract*)
Check if this EPG provides a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is provided by this EPG.

Returns True or False. True if the EPG does provide the Contract.

dont_consume(*contract*)
Make this EPG not consume a Contract. It does not check to see if the Contract was already consumed

Parameters **contract** – Instance of Contract class to be no longer consumed by this EPG.

Returns True

dont_consume_cif(*contract_interface*)
Make this EPG not consume a ContractInterface. It does not check to see if the ContractInterface was already consumed

Parameters **contract_interface** –

Returns True

dont_protect(*taboo*)
Make this EPG not protected by a Taboo

Parameters **taboo** – Instance of Taboo class to no longer protect this EPG.

Returns True

dont_provide(*contract*)
Make this EPG not provide a Contract

Parameters **contract** – Instance of Contract class to be no longer provided by this EPG.

Returns True

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *parent=None*, *tenant=None*)
Gets all of the EPGs from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Instance of the AppProfile class used to limit the EPGs retreived from the APIC.
- **tenant** – Instance of Tenant class used to limit the EPGs retreived from the APIC.

Returns List of CommonEPG instances (or EPG instances if called from EPG class)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consumed(*deleted=False*, *include_any_epg=False*)
Get all of the Contracts consumed by this EPG

Parameters

- **deleted** – Boolean indicating whether to get Contracts that are consumed or that the consumed was marked as deleted
- **include_any_epg** – Boolean indicating whether to include Contracts that are consumed due to inheritance from an AnyEPG within the same Context that this EPG is in.

Returns List of Contract objects that are consumed by the EPG.

get_all_consumed_cif(*deleted=False*)
Get all of the ContractInterfaces consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get ContractInterfaces that are consumed or that the consumed was marked as deleted

Returns List of ContractInterface objects that are consumed by the EPG.

get_all_protected(*deleted=False*)
Get all of the Taboos protecting this EPG

Parameters **deleted** – Boolean indicating whether to get Taboos that are protected or that the protected was marked as deleted

Returns List of Taboo objects that are protecting the EPG.

get_all_provided(*deleted=False*, *include_any_epg=False*)
Get all of the Contracts provided by this EPG

Parameters

- **deleted** – Boolean indicating whether to get Contracts that are provided or that the provided was marked as deleted
- **include_any_epg** – Boolean indicating whether to include Contracts that are provided due to inheritance from an AnyEPG within the same Context that this EPG is in.

Returns List of Contract objects that are provided by the EPG.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_bd()
Return the assigned BridgeDomain. There should only be one item in the returned list.

Returns List of BridgeDomain objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

static get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interfaces that this EPG is attached. The default is to get list of 'attached' interfaces. If 'status' is set to 'detached' it will return the list of detached Interface objects (Those EPGs which are no longer attached to an Interface, but the configuration is not yet pushed to the APIC.)

Parameters **status** – 'attached' or 'detached'. Defaults to 'attached'.

Returns List of Interface objects

get_json()
Returns json representation of the EPG

Returns json dictionary of the EPG

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*epgs*, *title=''*)
Will create table of EPG information for a given tenant

Parameters

- **epgs** –
- **title** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_bd()
Check if a BridgeDomain has been assigned to the EPG

Returns True or False. True if the EPG has been assigned a BridgeDomain.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_attributed_based
Get whether the EPG is attribute based :return: True if attribute based. False otherwise.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

protect(*taboo*)
Make this EPG protected by a Taboo

Parameters **taboo** – Instance of Taboo class to protect this EPG.

Returns True

provide(*contract*)
Make this EPG provide a Contract

Parameters **contract** – Instance of Contract class to be provided by this EPG.

Returns True

remove_bd()
Remove BridgeDomain from the EPG. Note that there should only be one BridgeDomain attached to the EPG.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_base_epg(*epg*)
Sets the Base EPG. Used by Attribute-based EPGs to indicate that the BridgeDomain, NodeAttach, and PathAttach relations should be copied from the base EPG when generating JSON.

Parameters **epg** – EPG class instance of the Base EPG

Returns None

set_deployment_immediacy(*immediacy*)
Set the deployment immediacy of the EPG

Parameters **immediacy** – String containing either "immediate" or "lazy"

set_dom_deployment_immediacy(*immediacy*)
Set the deployment immediacy for PhysDomain of the EPG

Parameters **immediacy** – String containing either "immediate" or "lazy"

set_dom_resolution_immediacy(*immediacy*)
Set the resolution immediacy for PhysDomain of the EPG

Parameters **immediacy** – String containing either "immediate" or "lazy"

set_intra_epg_isolation(*isolation*)
Set the intra-EPG isolation of the EPG

Parameters **isolation** – String containing either "unenforced" or "enforced"

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**EPGDomain**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

EPGDomain class

Parameters

- **name** – String containing the name of a source relation to an infrastructure domain profile associated with application endpoint groups. The domain profile can be either a VMM domain profile or a physical domain profile.
- **parent** – An instance of EPG class representing the EPG which contains this Domain Profile.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)
Gets all of the Physical Domains from the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns List of Switch Profile objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name(*session*, *infra_name*)
Gets all of the Physical Domains from the APIC

Parameters

- **infra_name** –
- **session** – the instance of Session used for APIC communication

Returns List of Switch Profile objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the fvTenant object

Returns A json dictionary of fvTenant

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Endpoint**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Endpoint class

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

static get(*session*, *endpoint_name=None*)
Gets all of the endpoints connected to the fabric from the APIC

Parameters

- **endpoint_name** –
- **session** – Session instance used to communicate with the APIC. Assumed to be logged in

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.

- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

classmethod get_all_by_epg(*session*, *tenant_name*, *app_name*, *epg_name*, *with_interface_attachments=True*)

Get all of the Endpoints for a specified EPG

Parameters

- **session** – Session instance used to communicate with the APIC. Assumed to be logged in
- **tenant_name** – String containing the tenant name
- **app_name** – String containing the app name
- **epg_name** – String containing the epg name
- **with_interface_attachments** – Boolean indicating whether interfaces should be attached or not. True is default.

Returns List of Endpoint instances

get_attributes(*name=None*)

Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)

Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)

Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)

Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –

- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*, *with_relations=True*)

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*endpoints*, *title=''*)
Will create table of taboo information for a given tenant

Parameters

- **title** –
- **endpoints** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**FexInterface**(*if_type*, *pod*, *node*, *fex*, *module*, *port*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This class describes a physical interface on a FEX device

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

classmethod is_dn_a_fex_interface(*dn*)

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

classmethod parse_dn(*dn*)

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Filter**(*filter_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Filter : roughly equivalent to vzFilter

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *tenant*)
Gets all of the Filters for the current tenant from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the Filters retreived from the APIC

Returns List of Filter objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name_and_tenant(*session*, *tenant*, *filter_name*)
Returns the Filter Object with name == filter_name and tenant == tenant

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant used to limit the search scope
- **filter_name** – the searched Filter Object

Returns a single Filter Object

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

static **get_from_json**(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the Filter

Returns json dictionary of the Filter

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_entry(*applyToFrag*, *arpOpc*, *dFromPort*, *dToPort*, *etherT*, *prot*, *sFromPort*, *sToPort*, *tcpRules*, *stateful*, *icmpv4T='not-given'*, *icmpv6T='not-given'*)
Returns whether or not the Filter has a FilterEntry. All fields are compared except name.

Returns True if the Filter has a matching FilterEntry. False otherwise

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**FilterEntry**(*name*, *parent*, *applyToFrag='0'*, *arpOpc='0'*, *dFromPort='0'*, *dToPort='0'*, *etherT='0'*, *prot='0'*, *sFromPort='0'*, *sToPort='0'*, *tcpRules='0'*, *stateful='0'*, *icmpv4T='not-given'*, *icmpv6T='not-given'*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

FilterEntry : roughly equivalent to vzEntry

Parameters

- **name** – String containing the name of this FilterEntry instance.
- **applyToFrag** – True or False. True indicates that this FilterEntry should be applied to IP fragments.
- **arpOpc** – 'req' or 'reply'. Indicates that this FilterEntry should be applied to ARP Requests or ARP replies.
- **dFromPort** – String containing the lower L4 destination port number of the L4 destination port number range.
- **dToPort** – String containing the upper L4 destination port number of the L4 destination port number range.
- **etherT** – String containing the EtherType of the frame to be matched by this FilterEntry.
- **prot** – String containing the L4 protocol number to be matched by this FilterEntry.
- **sFromPort** – String containing the lower L4 source port number of the L4 source port number range.
- **sToPort** – String containing the upper L4 source port number of the L4 source port number range.
- **tcpRules** – Bit mask consisting of the TCP flags to be matched by this FilterEntry.
- **stateful** – True or False. True indicates that this FilterEntry should monitor the TCP ACK bit.
- **icmpv4T** – String containing the ICMPv4 type.
- **icmpv6T** – String containing the ICMPv6 type.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

classmethod create_from_apic_json(*data*, *parent*)
create from the apic json

Parameters

- **data** – json dictionary

- **parent** – parent object

Returns object created from json dictionary

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent*, *tenant*)
To get all of acitoolkit style Filter Entries APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the FilterEntry

Returns json dictionary of the FilterEntry

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*filters*, *title=''*)
Will create table of filter information for a given tenant

Parameters

- **title** –
- **filters** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**IPEndpoint**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Endpoint class

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

static get(*session*)
Gets all of the IP endpoints connected to the fabric from the APIC

Parameters **session** – Session instance assumed to be logged into the APIC

Returns List of IPEndpoint instances

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

classmethod get_all_by_epg(*session*, *tenant_name*, *app_name*, *epg_name*)
Get all of the IP Endpoints for the specified EPG

Parameters

- **session** – Session instance assumed to be logged into the APIC
- **tenant_name** – String containing the Tenant name that holds the EPG
- **app_name** – String containing the AppProfile name that holds the EPG
- **epg_name** – String containing the EPG name

Returns List of IPEndpoint instances

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –

- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*)

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**InputTerminal**(*terminal_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.BaseTerminal*

Input terminal for a graph. It is input with respect to the provider

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_filter(*filter_obj*)
Add Filter to the Terminal, roughly equivalent to vzRsFiltAtt

Parameters **filter_obj** – Instance of Filter class. Represents a Filter that is added to the Terminal. Multiple Filters can be assigned to a single Terminal.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_filters()
Get all of the filters that are attached to this Terminal.

Returns List of Filter objects

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the Terminal Object

Returns json dictionary of the ContractSubject

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree (*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe (*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db (*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**L2ExtDomain** (*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

L2ExtDomain class

Parameters

- **name** – String containing the L2ExtDomain name
- **parent** – An instance of DomP class representing

add_child (*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag (*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach (*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session (*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)
Gets all of the L2Ext Domains from the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns List of L2ExtDomain objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name(*session*, *infra_name*)
Gets all of the Physical Domainss from the APIC

Parameters

- **infra_name** –
- **session** – the instance of Session used for APIC communication

Returns List of L2ExtDomain objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the l2extDomP object

Returns A json dictionary of fvTenant

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False

- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**L2Interface**(*name*, *encap_type*, *encap_id*, *encap_mode=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

The L2Interface class creates an logical L2 interface that can be attached to a physical interface. This interface defines the L2 encapsulation i.e. VLAN, VXLAN, or NVGRE

Parameters

- **name** – String containing the L2Interface instance name
- **encap_type** – String containing the encapsulation type. Valid values are 'VLAN', 'VXLAN', or 'NVGRE'.
- **encap_id** – String containing the encapsulation specific identifier representing the virtual L2 network (i.e. for VXLAN, this is the numeric value of the VNID).

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_encap_id()
Get the encap_id of the L2 interface. The value is returned as a string and depends on the encap_type (i.e. VLAN VID, VXLAN VNID, or NVGRE VSID)

Returns String containing encapsulation identifier value.

get_encap_type()
Get the encap_type of the L2 interface. Valid values are 'vlan', 'vxlan', and 'nvgre'

Returns String containing encap_type value.

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Returns whether this instance is considered an interface.

Returns True

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

static parse_encap(*encap*)
Parses the encap_type and encap_id from a json encap string Examples: vlan-515 / vxlan-5000

Parameters **encap** – String containing the json encap format

Returns encap_type, encap_id

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**L3ExtDomain**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

L3ExtDomain class

Parameters

- **name** – String containing the name of the external routed domain
- **parent** – An instance of DomP class

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)
Gets all of the Physical Domains from the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns List of L3Ext Domain objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name(*session*, *infra_name*)
Gets all of the L3Ext Domains from the APIC

Parameters

- **infra_name** –
- **session** – the instance of Session used for APIC communication

Returns List of L3Ext Domain objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –

- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the fvTenant object

Returns A json dictionary of fvTenant

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension='*')
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**L3Interface**(*name*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Creates an L3 interface that can be attached to an L2 interface. This interface defines the L3 address i.e. IPv4

Parameters **name** – String containing the name of this L3Interface object.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_context(*context*)
Add context to the EPG

Parameters **context** – Instance of Context class to assign to this L3Interface.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_addr()
Get the L3 address assigned to this interface. The address is set via the L3Interface.set_addr() method

Returns String containing the L3 address in dotted decimal notation.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_context()
Return the assigned context

Returns Instance of Context class that this L3Interface is assigned. If no Context is assigned, None is returned.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of L3Interface

Returns json dictionary of L3Interface

get_l3if_type()
Get the l3if_type of this L3 interface.

Returns L3 interface type. Valid values are 'sub-interface', 'l3-port', and 'ext-svi'

get_mtu()
Get the MTU of this interface

Returns MTU of the interface

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_context()
Check if the context has been assigned

Returns True or False. True if a Context has been assigned to this L3Interface.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Check if this is an interface object.

Returns True

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_context()
Remove context from the EPG

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_addr(*addr*)
Set the L3 address assigned to this interface

Parameters **addr** – String containing the L3 address in dotted decimal notation.

set_l3if_type(*l3if_type*)
Set the l3if_type of this L3 interface.

Parameters **l3if_type** – L3 interface type. Valid values are 'sub-interface', 'l3-port', and 'ext-svi'

set_mtu(*mtu*)
Set the L3 MTU of this interface

Parameters **mtu** – String containing MTU

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC

- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**LogicalModel**(*session=None*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This is the root class for the logical part of the network. Its corollary is the PhysicalModel class. It is a container that can hold all of logical model instances such as Tenants.

From this class, you can populate all of the children classes.

Initialization method that sets up the Fabric. :return:

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session=None*, *parent=None*)
Method to get all of the LogicalModels. It will get one and return it in a list.

Parameters

- **session** –
- **parent** –

Returns list of LogicalModel

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree (*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe (*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db (*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**MonitorPolicy** (*policyType*, *name*)
Bases: *acitoolkit.acitoolkit.BaseMonitorClass*

This class is the top-most container for a monitoring policy that controls how statistics are gathered. It has immediate children, CollectionPolicy objects, that control the default behavior for any network element that uses this monitoring policy. It may optionally have MonitorTarget objects as children that are used to override the default behavior for a particular target class such as Interfaces. There can be further granularity of control through children of the MonitorTarget sub-objects.

Children of the MonitorPolicy will be CollectionPolicy objects that define the collection policy plus optional MonitorTarget objects that allow finer grained control over specific target APIC objects such as 'l1PhysIf' (layer 1 physical interface).

The CollectionPolicy children are contained in a dictionary called "collection_policy" that is indexed by the granulariy of the CollectionPolicy, e.g. '5min', '15min', etc.

The MonitorTarget children are contained in a dictionary called "monitor_target" that is indexed by the name of the target object, e.g. 'l1PhysIf'.

To make a policy take effect for a particular port, for example, you must attach that monitoring policy to the port.

Note that the name of the MonitorPolicy is used to construct the dn of the object in the APIC. As a result, the name cannot be changed. If you read a policy from the APIC, change the name, and write it back, it will create a new policy with the new name and leave the old, original policy, in place with its original name.

A description may be optionally added to the policy.

The MonitorPolicy is initialized with simply a policy type and a name. There are two policy types: 'fabric' and 'access'. The 'fabric' monitoring policies can be applied to certain MonitorTarget types and 'access' monitoring policies can be applied to other MonitorTarget types. Initially however, both policies can have l1PhysIf as targets.

A name must be specified because it is used to build the distinguising name (dn) along with the policyType in the APIC. The dn for "fabric" policies will be /uni/fabric/monfabric-[name] and for "access" policies it will be /uni/infra/moninfra-[name] in the APIC.

Parameters

- **policyType** – String specifying whether this is a fabric or access policy
- **name** – String specifying a name for the policy.

add_collection_policy(*coll_obj*)
Add a collection policy.

Parameters **coll_obj** – A collection policy object of type CollectionPolicy

add_stats(*stat_obj*)
Adds a stats family object.

Parameters **stat_obj** – Statistics family object of type MonitorStats.

add_target(*target_obj*)
Add a target object.

Parameters **target_obj** – target object of type MonitorTarget

flat(*target='l1PhysIf'*)
This method will return a data structure that is a flattened version of the monitor policy. The flattened version is one that walks through the heirarchy of the policy and determines the administrative state and retention policy for each granularity of each statistics family. This is done for the target specified, i.e. 'l1PhysIf'

For example, if 'foo' is a MonitorPolicy object, then flatPol = foo.flat('l1PhysIf') will return a dictionary that looks like the following:

adminState = flatPol['counter_family']['granularity'].adminState retention = flatPol['counter_family']['granularity'].retention

The dictionary will have all of the counter families for all of the granularities and the value returned is the administrative state and retention value that is the final result of resolving the policy hierarchy.

Parameters **target** – APIC target object. This will default to 'l1PhysIf'

Returns Dictionary of statistic administrative state and retentions indexed by counter family and granularity.

classmethod get(*session*)

get() will get all of the monitor policies from the APIC and return them as a list. It will get both fabric and access (infra) policies including default policies.

Parameters **session** – the instance of Session used for APIC communication

Returns List of MonitorPolicy objects

get_parent()

Returns parent object

isModified()
Returns True if this policy and any children have been modified or created and not been written to the APIC

remove_collection_policy(*collection*)
Remove a collection_policy object. The object to remove is identified by its granularity, e.g. '5min', '15min', etc. This string can be found in the 'CollectionPolicy.granularity' attribute of the object.

Parameters **collection** – CollectionPolicy to remove.

remove_stats(*stats_family*)
Remove a stats family object. The object to remove is identified by a string, e.g. 'ingrPkts', or 'egrTotal'. This string can be found in the 'MonitorStats.scope' attribute of the object.

Parameters **stats_family** – Statistics family string.

remove_target(*target*)
Remove a target object. The object to remove is identified by a string, e.g 'l1PhysIf'. This string can be found in the 'MonitorTarget.scope' attribute of the object.

Parameters **target** – target to remove.

set_description(*description*)

Sets the description of the MonitorStats.

Parameters **description** – String to use as the description

set_name(*name*)

Sets the name of the MonitorStats.

Parameters **name** – String to use as the name

class acitoolkit.acitoolkit.**MonitorStats**(*parent*, *statsFamily*)
Bases: *acitoolkit.acitoolkit.BaseMonitorClass*

This class is a child of a MonitorTarget object. It is used to specify a scope for applying a monitoring policy that is more fine grained than the MonitorTarget. Specifically, the MonitorStats object specifies a statistics family such as "ingress packets" or "egress bytes".

The MonitorStats object must always be initialized with a parent object of type MonitorTarget. It sets the scope of its children collection policies (CollectionPolicy) to a particular statistics family.

The MonitorStats object contains a dictionary of collection policies called collection_policy. This is a dictionary of children CollectionPolicy objects indexed by their granularity, e.g. '5min', '15min', etc.

Parameters

- **parent** – Parent object that this monitor stats object should be applied to. This must be an object of type MonitorTarget.
- **statsFamily** – String specifying the statistics family that the children collection policies should be applied to. Possible values are:['egrBytes', 'egrPkts', 'egrTotal', 'egrDropPkts', 'ingrBytes', 'ingrPkts', 'ingrTotal', 'ingrDropPkts', 'ingrUnkBytes', 'ingrUnkPkts', 'ingrStorm']

add_collection_policy(*coll_obj*)
Add a collection policy.

Parameters **coll_obj** – A collection policy object of type CollectionPolicy

add_stats(*stat_obj*)
Adds a stats family object.

Parameters **stat_obj** – Statistics family object of type MonitorStats.

add_target(*target_obj*)
Add a target object.

Parameters **target_obj** – target object of type MonitorTarget

get_parent()

Returns parent object

isModified()
Returns True if this policy and any children have been modified or created and not been written to the APIC

remove_collection_policy(*collection*)
Remove a collection_policy object. The object to remove is identified by its granularity, e.g. '5min', '15min', etc. This string can be found in the 'CollectionPolicy.granularity' attribute of the object.

Parameters **collection** – CollectionPolicy to remove.

remove_stats(*stats_family*)
Remove a stats family object. The object to remove is identified by a string, e.g. 'ingrPkts', or 'egrTotal'. This string can be found in the 'MonitorStats.scope' attribute of the object.

Parameters **stats_family** – Statistics family string.

remove_target(*target*)
Remove a target object. The object to remove is identified by a string, e.g 'l1PhysIf'. This string can be found in the 'MonitorTarget.scope' attribute of the object.

Parameters **target** – target to remove.

set_description(*description*)

Sets the description of the MonitorStats.

Parameters **description** – String to use as the description

set_name(*name*)

Sets the name of the MonitorStats.

Parameters **name** – String to use as the name

statsDictionary = {'eqptIngrDropPkts': 'ingrDropPkts', 'eqptIngrBytes': 'ingrBytes', 'eqptEgrPkts': 'egrPkts', 'eq

statsFamilyEnum = ['egrBytes', 'egrPkts', 'egrTotal', 'egrDropPkts', 'ingrBytes', 'ingrPkts', 'ingrTotal', 'ingrDropPk

class acitoolkit.acitoolkit.**MonitorTarget**(*parent*, *target*)
Bases: *acitoolkit.acitoolkit.BaseMonitorClass*

This class is a child of a MonitorPolicy object. It is used to specify a scope for appling a monitoring policy. An example scope would be the Interface class, meaning that the monitoring policies specified here will apply to all Interface clas objects (l1PhysIf in the APIC) that use the parent MonitoringPolicy as their monitoring policy.

Children of the MonitorTarget will be CollectionPolicy objects that define the collection policy for the specified target plus optional MonitorStats objects that allow finer grained control over specific families of statistics such as ingress packets, ingrPkts.

The CollectionPolicy children are contained in a dictionary called "collection_policy" that is indexed by the granularity of the CollectionPolicy, e.g. '5min', '15min', etc.

The MonitorStats children are contained in a dictionary called "monitor_stats" that is indexed by the name of the statistics family, e.g. 'ingrBytes', 'ingrPkts', etc.

The MonitorTarget object is initialized with a parent of type MonitorPolicy, and a target string. Initially, this toolkit only supports a target of type 'l1PhysIf'. The 'l1PhyIf' target is a layer 1 physical interface or "port". The MonitorTarget will narrow the scope of the policy specified by the children of the MonitorTarget to be only the target class.

Parameters

- **parent** – Parent object that this monitor target is a child. It must be of type MonitorPolicy
- **target** – String specifying the target class for the Monitor policy.

add_collection_policy(*coll_obj*)
Add a collection policy.

Parameters **coll_obj** – A collection policy object of type CollectionPolicy

add_stats(*stat_obj*)
Adds a stats family object.

Parameters **stat_obj** – Statistics family object of type MonitorStats.

add_target(*target_obj*)
Add a target object.

Parameters **target_obj** – target object of type MonitorTarget

get_parent()

Returns parent object

isModified()
Returns True if this policy and any children have been modified or created and not been written to the APIC

remove_collection_policy(*collection*)
Remove a collection_policy object. The object to remove is identified by its granularity, e.g. '5min', '15min', etc. This string can be found in the 'CollectionPolicy.granularity' attribute of the object.

Parameters **collection** – CollectionPolicy to remove.

remove_stats(*stats_family*)
Remove a stats family object. The object to remove is identified by a string, e.g. 'ingrPkts', or 'egrTotal'. This string can be found in the 'MonitorStats.scope' attribute of the object.

Parameters **stats_family** – Statistics family string.

remove_target(*target*)
Remove a target object. The object to remove is identified by a string, e.g 'l1PhysIf'. This string can be found in the 'MonitorTarget.scope' attribute of the object.

Parameters **target** – target to remove.

set_description(*description*)

Sets the description of the MonitorStats.

Parameters **description** – String to use as the description

set_name(*name*)

Sets the name of the MonitorStats.

Parameters **name** – String to use as the name

class acitoolkit.acitoolkit.**NetworkPool**(*name*, *encap_type*, *start_id*, *end_id*, *mode*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This class defines a pool of network ids

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

static get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no format parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the format string appended.

Parameters **fmt** – optional format string, default is 'json'

Returns URL string

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree (*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe (*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db (*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OSPFInterface** (*name*, *router=None*, *area_id=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Creates an OSPF router interface that can be attached to a L3 interface. This interface defines the OSPF area, authentication, etc.

Parameters

- **name** – String containing the name of this OSPFInterface object.
- **area_id** – String containing the OSPF area id of this interface. Default is None.

add_child (*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag (*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach (*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session (*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of OSPFInterface :returns: json dictionary of OSPFInterface

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Returns whether this instance is considered an interface. :returns: True

static is_ospf()

Returns True if this interface is an OSPF interface. In the case of OSPFInterface instances, this is always True.

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_area_type(*area_type*)
Set the area_type for this OSPFInterface

Parameters **area_type** – AreaType to use for this OSPFInterface

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes

- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OSPFInterfacePolicy**(*name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Represents the interface settings of an OSPF interface

param name: String containing the name of this OSPF interface policy param parent: Instance of the Tenant class representing the tenant owning this OSPF interface policy

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of OSPFRouter

Returns json dictionary of OSPFIRouter

get_nw_type()
Get the nw-type of this interface ospf policy :returns: string of the network type for this policy

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_nw_type(*network_type*)
sets the L3 nw_type with some validation

Parameters **network_type** – string of bcast or p2p

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OSPFRouter**(*name*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Represents the global settings of the OSPF Router

Parameters **name** – String containing the name of this OSPFRouter object.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of OSPFRouter

Returns json dictionary of OSPFIRouter

get_node_id()
:returns string containing the Node ID

get_parent()

Returns Parent of this object.

get_router_id()
:returns string containing the Router ID

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_node_id(*node*)
Sets the router id of the object

Parameters **node** – String containing the node id

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_router_id(*rid*)
Sets the router id of the object

Parameters **rid** – String containing the router id

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutputTerminal**(*terminal_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.BaseTerminal*

Input terminal for a graph. It is input with respect to the provider

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_filter(*filter_obj*)
Add Filter to the Terminal, roughly equivalent to vzRsFiltAtt

Parameters **filter_obj** – Instance of Filter class. Represents a Filter that is added to the Terminal. Multiple Filters can be assigned to a single Terminal.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_filters()
Get all of the filters that are attached to this Terminal.

Returns List of Filter objects

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the Terminal Object

Returns json dictionary of the ContractSubject

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutsideEPG**(*epg_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.CommonEPG*

OutsideEPG class, roughly equivalent to l3ext:InstP

Parameters

- **epg_name** – String containing the name of this EPG
- **parent** – Instance of the AppProfile class representing the Application Profile where this EPG is contained.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

consume(*contract*)
Make this EPG consume a Contract

Parameters **contract** – Contract class instance to be consumed by this EPG.

Returns True

consume_cif(*contract_interface*)
Make this EPG consume a ContractInterface

Parameters **contract_interface** – ContractInterface class instance to be consumed by this EPG.

Returns True

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_consume(*contract*)
Check if this EPG consumes a specific Contract

Parameters **contract** – Instance of Contract class to check if it is consumed by this EPG.

Returns True or False. True if the EPG does consume the Contract.

does_consume_cif(*contract_interface*)
Check if this EPG consumes a specific Contract

Parameters **contract_interface** –

Returns True or False. True if the EPG does consume the ContractInterface.

does_protect(*taboo*)
Check if this EPG is protected by a specific Taboo.

Parameters **taboo** – Instance of Taboo class to check if it protects this EPG.

Returns True or False. True if the EPG is protected by the Taboo.

does_provide(*contract*)
Check if this EPG provides a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is provided by this EPG.

Returns True or False. True if the EPG does provide the Contract.

dont_consume(*contract*)
Make this EPG not consume a Contract. It does not check to see if the Contract was already consumed

Parameters **contract** – Instance of Contract class to be no longer consumed by this EPG.

Returns True

dont_consume_cif(*contract_interface*)
Make this EPG not consume a ContractInterface. It does not check to see if the ContractInterface was already consumed

Parameters **contract_interface** –

Returns True

dont_protect(*taboo*)
Make this EPG not protected by a Taboo

Parameters **taboo** – Instance of Taboo class to no longer protect this EPG.

Returns True

dont_provide(*contract*)
Make this EPG not provide a Contract

Parameters **contract** – Instance of Contract class to be no longer provided by this EPG.

Returns True

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *parent=None*, *tenant=None*)
Gets all of the EPGs from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Instance of the AppProfile class used to limit the EPGs retreived from the APIC.
- **tenant** – Instance of Tenant class used to limit the EPGs retreived from the APIC.

Returns List of CommonEPG instances (or EPG instances if called from EPG class)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consumed(*deleted=False*)
Get all of the Contracts consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are consumed or that the consumed was marked as deleted

Returns List of Contract objects that are consumed by the EPG.

get_all_consumed_cif(*deleted=False*)
Get all of the ContractInterfaces consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get ContractInterfaces that are consumed or that the consumed was marked as deleted

Returns List of ContractInterface objects that are consumed by the EPG.

get_all_protected(*deleted=False*)
Get all of the Taboos protecting this EPG

Parameters **deleted** – Boolean indicating whether to get Taboos that are protected or that the protected was marked as deleted

Returns List of Taboo objects that are protecting the EPG.

get_all_provided(*deleted=False*)
Get all of the Contracts provided by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are provided or that the provided was marked as deleted

Returns List of Contract objects that are provided by the EPG.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interfaces that this EPG is attached. The default is to get list of 'attached' interfaces. If 'status' is set to 'detached' it will return the list of detached Interface objects (Those EPGs which are no longer attached to an Interface, but the configuration is not yet pushed to the APIC.)

Parameters **status** – 'attached' or 'detached'. Defaults to 'attached'.

Returns List of Interface objects

get_json()
Returns json representation of the EPG

Returns json dictionary of the EPG

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

protect(*taboo*)
Make this EPG protected by a Taboo

Parameters **taboo** – Instance of Taboo class to protect this EPG.

Returns True

provide(*contract*)
Make this EPG provide a Contract

Parameters **contract** – Instance of Contract class to be provided by this EPG.

Returns True

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutsideL2**(*l2out_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Represents the L2Out for external connectivity

Parameters

- **l2out_name** – String containing the name of this OutsideL2
- **parent** – Instance of the Tenant class representing the tenant owning this OutsideL2.

add_bd(*bd*)
Add BridgeDomain to the EPG

Parameters **bd** – Instance of BridgeDomain class to assign to this OutsideL2.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_l2extdom(*extdom*)
Set the L2ExternalDomain for this BD

Parameters **extdom** –

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.

- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –

- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of OutsideL2

Returns json dictionary of OutsideL2

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_bd()
Check if the BridgeDomain has been assigned

Returns True or False. True if a BridgeDomain has been assigned to this L2Interface.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_l2extdom()

Returns Boolean indicating presence of L2 External Domain Attachment

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_bd()
Remove the BridgeDomain from the EPG

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutsideL2EPG**(*epg_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.CommonEPG*

OutsideL2EPG class, roughly equivalent to l2ext:InstP

Parameters

- **epg_name** – String containing the name of this EPG
- **parent** – Instance of the AppProfile class representing the Application Profile where this EPG is contained.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

consume(*contract*)
Make this EPG consume a Contract

Parameters **contract** – Contract class instance to be consumed by this EPG.

Returns True

consume_cif(*contract_interface*)
Make this EPG consume a ContractInterface

Parameters **contract_interface** – ContractInterface class instance to be consumed by this EPG.

Returns True

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

does_consume(*contract*)
Check if this EPG consumes a specific Contract

Parameters **contract** – Instance of Contract class to check if it is consumed by this EPG.

Returns True or False. True if the EPG does consume the Contract.

does_consume_cif(*contract_interface*)
Check if this EPG consumes a specific Contract

Parameters **contract_interface** –

Returns True or False. True if the EPG does consume the ContractInterface.

does_protect(*taboo*)
Check if this EPG is protected by a specific Taboo.

Parameters **taboo** – Instance of Taboo class to check if it protects this EPG.

Returns True or False. True if the EPG is protected by the Taboo.

does_provide(*contract*)
Check if this EPG provides a specific Contract.

Parameters **contract** – Instance of Contract class to check if it is provided by this EPG.

Returns True or False. True if the EPG does provide the Contract.

dont_consume(*contract*)
Make this EPG not consume a Contract. It does not check to see if the Contract was already consumed

Parameters **contract** – Instance of Contract class to be no longer consumed by this EPG.

Returns True

dont_consume_cif(*contract_interface*)
Make this EPG not consume a ContractInterface. It does not check to see if the ContractInterface was already consumed

Parameters **contract_interface** –

Returns True

dont_protect(*taboo*)
Make this EPG not protected by a Taboo

Parameters **taboo** – Instance of Taboo class to no longer protect this EPG.

Returns True

dont_provide(*contract*)
Make this EPG not provide a Contract

Parameters **contract** – Instance of Contract class to be no longer provided by this EPG.

Returns True

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *parent=None*, *tenant=None*)
Gets all of the EPGs from the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **parent** – Instance of the AppProfile class used to limit the EPGs retreived from the APIC.
- **tenant** – Instance of Tenant class used to limit the EPGs retreived from the APIC.

Returns List of CommonEPG instances (or EPG instances if called from EPG class)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_consumed(*deleted=False*)
Get all of the Contracts consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are consumed or that the consumed was marked as deleted

Returns List of Contract objects that are consumed by the EPG.

get_all_consumed_cif(*deleted=False*)
Get all of the ContractInterfaces consumed by this EPG

Parameters **deleted** – Boolean indicating whether to get ContractInterfaces that are consumed or that the consumed was marked as deleted

Returns List of ContractInterface objects that are consumed by the EPG.

get_all_protected(*deleted=False*)
Get all of the Taboos protecting this EPG

Parameters **deleted** – Boolean indicating whether to get Taboos that are protected or that the protected was marked as deleted

Returns List of Taboo objects that are protecting the EPG.

get_all_provided(*deleted=False*)
Get all of the Contracts provided by this EPG

Parameters **deleted** – Boolean indicating whether to get Contracts that are provided or that the provided was marked as deleted

Returns List of Contract objects that are provided by the EPG.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interfaces that this EPG is attached. The default is to get list of 'attached' interfaces. If 'status' is set to 'detached' it will return the list of detached Interface objects (Those EPGs which are no longer attached to an Interface, but the configuration is not yet pushed to the APIC.)

Parameters **status** – 'attached' or 'detached'. Defaults to 'attached'.

Returns List of Interface objects

get_json()
Returns json representation of the EPG

Returns json dictionary of the EPG

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

protect(*taboo*)
Make this EPG protected by a Taboo

Parameters **taboo** – Instance of Taboo class to protect this EPG.

Returns True

provide(*contract*)
Make this EPG provide a Contract

Parameters **contract** – Instance of Contract class to be provided by this EPG.

Returns True

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutsideL3**(*l3out_name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Represents the L3Out for external connectivity

Parameters

- **l3out_name** – String containing the name of this OutsideL3
- **parent** – Instance of the Tenant class representing the tenant owning this OutsideL3.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_context(*context*)
Add context to the EPG

Parameters **context** – Instance of Context class to assign to this OutsideL3.

add_l3extdom(*extdom*)
Set the L3ExternalDomain for this BD

Parameters **extdom** –

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_context()
Return the assigned context

Returns Instance of Context class that this OutsideL3 is assigned. If no Context is assigned, None is returned.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –

- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of OutsideL3

Returns json dictionary of OutsideL3

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_context()
Check if the context has been assigned

Returns True or False. True if a Context has been assigned to this L3Interface.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_l3extdom()

Returns Boolean indicating presence of L3 External Domain Attachment

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_context()
Remove the context from the EPG

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**OutsideNetwork**(*name*, *parent*)
Bases: *acitoolkit.acitoolkit.BaseSubnet*

OutsideNetwork class, roughly equivalent to l3extSubnet in the APIC model

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

addr
Subnet address

Returns String containing the subnet default gateway IP address and mask e.g. "1.2.3.4/24"

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_addr()
Get the subnet address

Returns The subnet address as a string in the form of <ipaddr>/<mask>

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the OutsideNetwork object.

Returns json dictionary of OutsideNetwork

get_parent()

Returns Parent of this object.

get_scope()
Get the subnet scope

Returns The subnet scope as a string

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

ip
IP address of the subnet in the form of Address/mask e.g. 10.1.1.1/16

Returns String containing the IP address

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_addr(*addr*)
Set the subnet address

Parameters **addr** – The subnet default gateway address as a string in the form of <ipaddr>/<mask>

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the subnet scope

Parameters **scope** – String containing the scope

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**PhysDomain**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

Physical Network domain

Parameters

- **name** – String containing the PhysDomain name
- **parent** – An instance of DomP class representing

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)
Gets all of the Physical Domains from the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns List of PhysDomain objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name(*session*, *infra_name*)
Gets all of the Physical Domains from the APIC

Parameters

- **infra_name** –
- **session** – the instance of Session used for APIC communication

Returns List of PhysDomain objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –

- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the fvTenant object

Returns A json dictionary of fvTenant

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

static get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no format parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the format string appended.

Parameters **fmt** – optional format string, default is 'json'

Returns URL string

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

push_to_apic(*session*)
Push the appropriate configuration to the APIC for this Phys Domain. All of the subobject configuration will also be pushed.

Parameters **session** – the instance of Session used for APIC communication

Returns Requests Response code

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**PortChannel**(*name*)
Bases: *acitoolkit.acibaseobject.BaseInterface*

This class defines a port channel interface.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*interface*)
Attach an interface to this PortChannel

Parameters **interface** –

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

classmethod **create_from_dn**(*dn*)
Create a PortChannel instance based on the specified DN

Parameters **dn** – String containing the DN

Returns Instance of PortChannel class

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*interface*)
Detach an interface from this PortChannel

Parameters **interface** –

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

static **get**(*session*)
Gets all of the port channel interfaces from the APIC

Parameters **session** –

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

Returns json representation of the PortChannel

Returns json dictionary of the PortChannel

get_parent()

Returns Parent of this object.

get_port_channel_selector_json(*port_name*)
Get the JSON for the Port Channel selector

Parameters **port_name** – String containing the port name

Returns Dictonary containing the JSON for the Port Channel selector

get_port_selector_json()
Returns the port selector.

Returns

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

static get_url(*fmt='json'*)
Get the URLs used to push the configuration to the APIC if no format parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the format string appended.

Parameters **fmt** – optional format string, default is 'json'

Returns URL string

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_dn_vpc(*dn*)
Check if the DN is a VPC

Parameters **dn** – String containing the DN

Returns True if the the DN is a VPC. False otherwise.

is_interface()
Returns True since a PortChannel is an interface

is_vpc()
Returns True if the PortChannel is a VPC

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL

- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Search**
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This is an empty class used to create a search object for use with the "find" method.

Attaching attributes to this class and then invoking find will return all objects with matching attributes in the object hierarchy at and below where the find is invoked.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Subnet**(*subnet_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.BaseSubnet*

Subnet : roughly equivalent to fvSubnet

Parameters

- **subnet_name** – String containing the name of this Subnet instance.
- **parent** – An instance of BridgeDomain class representing the BridgeDomain which contains this Subnet.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

addr
Subnet address

Returns String containing the subnet default gateway IP address and mask e.g. "1.2.3.4/24"

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *bridgedomain*, *tenant*)
Gets all of the Subnets from the APIC for a particular tenant and bridgedomain.

Parameters

- **session** – the instance of Session used for APIC communication
- **bridgedomain** – the instance of BridgeDomain used to limit the Subnet instances retreived from the APIC

- **tenant** – the instance of Tenant used to limit the Subnet instances retreived from the APIC

Returns List of Subnet objects

get_addr()
Get the subnet address

Returns The subnet address as a string in the form of <ipaddr>/<mask>

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

classmethod get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the subnet

Returns json dictionary of subnet

get_parent()

Returns Parent of this object.

get_scope()
Get the subnet scope

Returns The subnet scope as a string

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

ip
IP address of the subnet in the form of Address/mask e.g. 10.1.1.1/16

Returns String containing the IP address

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_addr(*addr*)
Set the subnet address

Parameters **addr** – The subnet default gateway address as a string in the form of <ipaddr>/<mask>

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the subnet scope

Parameters **scope** – The scope of the subnet. Use "public" when the subnet

needs to be advertised externally, "private" when no external routing for the subnet is required (only internal), and "shared" when a route for the subnet needs to be leaked to a different VRF within the fabric. Note that "public" and "private" are mutually exclusive, but "shared" can be appended to any of them ("e.g. set_scope("public,shared")).

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Taboo**(*contract_name*, *parent=None*)
Bases: *acitoolkit.acitoolkit.BaseContract*

Taboo : Class for Taboos

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_filter_entries(*direction='bidirectional-only'*)
Get all of the filter entries contained within this Contract/Taboo

Parameters **direction** – String containing the type of filter entries to gather Valid values are 'bidirectional-only', 'input-only', 'output-only', 'all' Default is 'bidirectional-only'

Returns List of FilterEntry instances

get_attributes(*name=None*)

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the contract

Returns json dictionary of the contract

get_parent()

Returns Parent of this object.

get_scope()
Get the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*taboos*, *title=''*)
Will create table of taboo information for a given tenant

Parameters

- **title** –
- **taboos** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

set_scope(*scope*)
Set the scope of this contract. Valid values are 'context', 'global', 'tenant', and 'application-profile'

Parameters **scope** – String containing one of the following 'context', 'global', 'tenant', or 'application-profile'

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Tag**(*name=None*, *parent=None*)
Bases: acitoolkit.acibaseobject._Tag

Tag class.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –

- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json(*obj_class*, *attributes=None*, *children=None*, *get_children=True*)
Get the JSON representation of this class in the actual APIC Object Model.

Parameters

- **obj_class** – Object Class Name within the APIC model.
- **attributes** – Additional attributes that should be set in the JSON.
- **children** – Children objects to traverse as well.
- **get_children** – Indicates whether the children objects should be included.

Returns JSON dictionary to be pushed to the APIC.

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the

currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**Tenant**(*name*, *parent=None*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

The Tenant class is used to represent the tenants within the acitoolkit object model. In the APIC model, this class is roughly equivalent to the fvTenant class.

Parameters

- **name** – String containing the Tenant name
- **parent** – None or An instance of Fabric class representing the Pod which contains this Tenant.

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

classmethod exists(*session*, *tenant*)
Check if a tenant exists on the APIC.

Parameters

- **session** – the instance of Session used for APIC communication
- **tenant** – the instance of Tenant to check if exists on the APIC

Returns True or False

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*, *parent=None*)
Gets all of the tenants from the APIC.

Parameters

- **parent** – Parent object of the Tenant
- **session** – the instance of Session used for APIC communication

Returns a list of Tenant objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

classmethod get_deep(*session*, *names=()*, *limit_to=()*, *subtree='full'*, *config_only=False*, *parent=None*)
Get the Tenant objects and all of the children objects.

Parameters

- **session** – the instance of Session used for APIC communication
- **names** – list of strings containing the tenant names. If no list is given, all tenants will be collected. It should be noted that if relations extend across tenants, the relation will only be populated if the tenants are included in this list.
- **limit_to** – list of strings containing the APIC classes to limit the collection to i.e. ['fvTenant', 'fvBD']. If no list is given, all classes will be collected.
- **subtree** – String containing the rsp-subtree option. Default is 'full'.
- **config_only** – Boolean containing whether to collect only configurable parameters
- **parent** – The parent instance to assign to the tenant objects. If None, a Fabric instance will be created.

Returns Requests Response code

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the fvTenant object

Returns A json dictionary of fvTenant

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

static get_table(*tenants*, *title=''*)
Will create table of switch context information

Parameters

- **title** –
- **tenants** –

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

static get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no format parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the format string appended.

Parameters **fmt** – optional format string, default is 'json'

Returns URL string

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

push_to_apic(*session*)
Push the appropriate configuration to the APIC for this Tenant. All of the subobject configuration will also be pushed.

Parameters **session** – the instance of Session used for APIC communication

Returns Requests Response code

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**TunnelInterface**(*if_type*, *pod*, *node*, *tunnel*)
Bases: object

This class describes a tunnel interface

class acitoolkit.acitoolkit.**VMM**(*name*, *ipaddr*, *credentials*, *vswitch_info*, *network_pool*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This class defines an instance of connectivity to a Virtual Machine Manager (such as VMware vCenter)

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with search_object and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of self are compared to all attributes of *search_object*. If search_object.<attr> exists and is the same as self.<attr> or search_object.<attr> is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –

- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

static get_url(*fmt='json'*)
Get the URL used to push the configuration to the APIC if no format parameter is specified, the format will be 'json' otherwise it will return '/api/mo/uni.' with the format string appended.

Parameters **fmt** – optional format string, default is 'json'

Returns URL string

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**VMMCredentials**(*name*, *uid*, *pwd*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

This class defines the credentials used to login to a Virtual Machine Manager

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

get(*session*, *toolkit_class*, *apic_class*, *parent=None*, *tenant=None*)
Generic classmethod to get all of a particular APIC class.

Parameters

- **session** – the instance of Session used for APIC communication
- **toolkit_class** – acitoolkit class to return
- **apic_class** – String containing class name from the APIC object model.
- **parent** – Object to assign as the parent to the created objects.
- **tenant** – Tenant object to assign the created objects.

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

class acitoolkit.acitoolkit.**VMMvSwitchInfo**(*vendor*, *container_name*, *vswitch_name*)
Bases: object

This class contains the information necessary for creating the vSwitch on the Virtual Machine Manager

class acitoolkit.acitoolkit.**VmmDomain**(*name*, *parent*)
Bases: *acitoolkit.acibaseobject.BaseACIObject*

VMMDomain class

Parameters

- **name** – String containing the VMM Domain name
- **parent** – An instance of DomP class

add_child(*obj*)
Add a child to the children list.

Parameters **obj** – Child object to add to the children list of the called object.

add_tag(*tag*)
Assign this object a particular tag. Tags are strings that can be used to classify objects. More than 1 tag can be assigned to an object.

Parameters **tag** – string containing the tag to assign to this object or an instance of _Tag

attach(*item*)
Attach the object to the other object.

Parameters **item** – Object to be attached.

check_session(*session*)
This will check that the session is of type Session and raise exception if it not

Parameters **session** – the session to check

Returns

delete_tag(*tag*)
Mark a particular tag as being deleted from this object.

Parameters **tag** – string containing the tag to delete from this object or an instance of _Tag

detach(*item*)
Detach the object from the other object. A relationship is either 'attached', 'detached', or does not exist. A detached relationship will cause the relationship to be deleted when pushed to the APIC.

Parameters **item** – Object to be detached.

find(*search_object*)
This will check to see if self is a match with `search_object` and then call find on all of the children of search. If there is a match, a list containing self and any matches found by the children will be returned as a list.

The criteria for a match is that all attributes of `self` are compared to all attributes of *search_object*. If `search_object.<attr>` exists and is the same as `self.<attr>` or `search_object.<attr>` is 'None', then that attribute matches. If all such attributes match, then there is a match and self will be returned in the result.

If there is an attribute of `search_object` that does not exist in `self`, it will be considered a mismatch. If there is an attribute of `self` that does not exist in `search_object`, it will be ignored.

Parameters **search_object** – ACI object to search

Returns List of objects

classmethod get(*session*)
Gets all of the VMM Domains from the APIC

Parameters **session** – the instance of Session used for APIC communication

Returns List of VMM Domain objects

get_all_attached(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the relations of objects belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_all_attachments(*attached_class*, *status='attached'*, *relation_type=None*)
Get all of the attachments to an object belonging to the specified class with the specified status.

Parameters

- **attached_class** – The class that is the subject of the search.
- **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

get_attributes(*name=None*)
Will return the value of the named attribute in a dictionary format. If no name is given, then it will return all attributes.

Note that attributes that start with _ (underbar) will NOT be included unless explicitly named

This method should be over-written as appropriate by inheriting objects to handle how their local attributes are implemented.

This is intended to normalize how all attributes on all objects can be accessed since the implementations were not consistent.

Parameters **name** – optional name of attribute to return

Returns dictionary of attributes and their values

classmethod get_by_name(*session*, *infra_name*)
Gets all of the VMM Domains from the APIC

Parameters

- **infra_name** –
- **session** – the instance of Session used for APIC communication

Returns List of VMM Domain objects

get_child(*child_type*, *child_name*)
Gets a specific immediate child of this object

Parameters

- **child_type** – Class of the child to return
- **child_name** – Name of the child to return

Returns The specific instance of child_type or None if not found

get_children(*only_class=None*)
Get a list of the immediate child objects of this object.

Parameters **only_class** – Optional parameter that will be used to limit the objects returned to only those belonging to the class passed in this parameter.

Returns List of children objects.

get_deep(*full_data*, *working_data*, *parent=None*, *limit_to=()*, *subtree='full'*, *config_only=False*)
Gets all instances of this class from the APIC and gets all of the children as well.

Parameters

- **full_data** –
- **working_data** –
- **parent** –
- **limit_to** –
- **subtree** –
- **config_only** –

get_deep_apic_classes(*include_concrete=False*)
Get all the apic classes needed for this acitoolkit class and all of its children. :return: list of all apic classes

get_dn_from_attributes(*attributes*)
Will get the dn from the attributes or construct it using the dn of the parent plus the rn. Failing those, it will return None

Parameters **attributes** –

Returns String containing dn or None

get_event(*session*)
Gets the event that is pending for this class. Events are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_fault(*session*, *extension=''*)
Gets the fault that is pending for this class. Faults are returned in the form of objects. Objects that have been deleted are marked as such.

Parameters **session** – the instance of Session used for APIC communication

get_from_json(*data*, *parent=None*)
returns a Tenant object from a json

get_interfaces(*status='attached'*)
Get all of the interface relations. Note that multiple classes are considered "interfaces" such as Interface, L2Interface, L3Interface, etc.

Parameters **status** – Valid values are 'attached' and 'detached'. Default is 'attached'.

Returns List of interfaces that this object has relations and the status matches.

get_json()
Returns json representation of the vmmDomP object

Returns A json dictionary of vmmDomP

get_parent()

Returns Parent of this object.

get_searchable()
Method to recursively retrieve all of the searchable items from all the children objects, add the current object to them as additional context, append the local searchable terms, and return the result.

get_table(*aci_object*, *title=''*)
Abstract method that should be replaced by a version that is specific to the object

Parameters

- **aci_object** –
- **title** – String containing the table title

Returns list of Table objects

get_tags()
Get the tags assigned to this object.

Returns List of tag instances

has_attachment(*item*)
Indicates whether this object is attached to the item/ :returns: True or False, True indicates the object is attached.

has_child(*obj*)
Check for existence of a child in the children list

Parameters **obj** – Child object that is the subject of the check.

Returns True or False, True indicates that it does indeed have the *obj* object as a child.

has_detachment(*item*)
Indicates whether the object is detached from this item. :returns: True or False, True indicates the object is detached.

has_events(*session*, *extension=''*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

Returns True or False. True if there are events pending.

has_parent()
returns True if this object has a parent

Returns bool

has_tag(*tag*)
Checks whether this object has a particular tag assigned.

Parameters **tag** – string containing the tag name or an instance of _Tag

Returns True or False. True indicates the object has this tag assigned.

has_tags()
Checks whether this object has any tags assigned at all.

Returns True or False. True indicates the object has at least one tag assigned.

info()
Node information summary.

Returns Formatted string that has a summary of all of the info gathered about the node.

infoList()
Node information. Returns a list of (attr, value) tuples.

Returns list of [(attr, value),]

is_attached(*item*)
Indicates whether the item is attached to this object/ :returns: True or False, True indicates the item is attached.

is_deleted()
Check if the object has been deleted.

Returns True or False, True indicates the object has been deleted.

is_detached(*item*)
Indicates whether the item is detached from this object.

Returns True or False, True indicates the item is detached.

is_interface()
Indicates whether this object is considered an Interface. The default is False.

Returns False

mark_as_deleted()
Mark the object as deleted. This will cause the JSON status to be set to deleted.

mask_class_from_graphs()
Mask (hide) this class from graph creation

Returns False indicating that this class should not be masked.

populate_children(*deep=False*, *include_concrete=False*)
Populates all of the children and then calls populate_children of those children if deep is True. This method should be overridden by any object that does have children.

If include_concrete is True, then if the object has concrete objects below it, i.e. is a switch, then also populate those conrete object.

Parameters

- **include_concrete** – True or False. Default is False
- **deep** – True or False. Default is False.

remove_child(*obj*)
Remove a child from the children list

Parameters **obj** – Child object that is to be removed.

remove_tag(*tag*)
Remove a particular tag from being assigned to this object. Note that this does not delete the tag from the APIC.

Parameters **tag** – string containing the tag to remove from this object or an instance of _Tag

set_parent(*parent_obj*)
Set the parent object

Parameters **parent_obj** – Instance of the parent object

Returns None

subscribe(*session*, *extension=''*, *only_new=False*)
Subscribe to events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

subscribe_to_fault_instances_subtree(*session*, *extension=''*, *deep=False*)
Subscribe to faults instances for the whole subtree.

Parameters

- **session** – the instance of Session used for APIC communication
- **extension** – Optional string that can be used to extend the URL
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

unsubscribe(*session*)
Unsubscribe for events from the APIC that pertain to instances of this class.

Parameters **session** – the instance of Session used for APIC communication

update_db(*session*, *subscribed_classes*, *deep=False*)

Parameters

- **session** – Session class instance representing the connection to the APIC
- **subscribed_classes** – List of subscribed classes
- **deep** – Boolean indicating whether to go deep or not. Default is False

Returns List of subscribed classes

acitoolkit.acitoolkit.**build_object_dictionary**(*objs*)

Will build a dictionary indexed by object class that contains all the objects of that class

Parameters **objs** –

Returns

acitoolkitlib module

Collection of utility classes to make getting credentials and configuration easier.

class acitoolkit.acitoolkitlib.**AcitoolkitGraphBuilder**

Bases: object

Class to build class hierarchy diagrams for the ACI toolkit Physical and Logical Models

static build_graph_from_parent(*root_parent_name*)

Create a graph starting from the root class name

Parameters **root_parent_name** – String containing the class name to use as the root of the class hierarchy graph

Returns None

build_graphs()

Build the graphs starting with the various parent class names

class acitoolkit.acitoolkitlib.**Credentials**(*qualifier='apic'*, *description=''*)

Bases: object

Used to get the APIC and MySQL login credentials from the command line (–help gives usage).

The login credentials are taken in the following order

•Command line arguments

•Environment variables

•File named credentials.py

•From an interactive prompt

These are done in a per credential basis so it is possible to specify only some of the arguments. For instance, the username and URL can be specified in credentials.py but the password can be taken from the user through the interactive prompt. Another example is using the command line argument to override the URL specified in credentials.py to temporarily connect to a different APIC.

add_argument(**args*, ***kwargs*)

Pass through function to allow the underlying parser to be extended.

add_argument_group(**args*, ***kwargs*)

Pass through function to allow the underlying parser to be extended.

add_mutually_exclusive_group(**args*, ***kwargs*)
Pass through function to allow the underlying parser to be extended.

get()
Get the arguments and verify them

print_help(**args*, ***kwargs*)
Pass through function to allow the underlying parser to be extended.

verify()
Verify that the arguments have been passed in some way. If not, ask the user through interactive prompt.

aciFaults module

This module deals with Fault objects.

class acitoolkit.aciFaults.**Faults**
Bases: *acitoolkit.acibaseobject.BaseACIObject*

A class for Fault objects

classmethod get_fault(*session*, *extension=''*)
Not implemented for this class. Use get_faults() instead

Parameters

- **session** – Not used
- **extension** – Not used

Raises AttributeError

classmethod get_faults(*session*, *fault_filter=None*, *tenant_name=None*)
Gets the fault that is pending for this class. Faults are returned in the form of objects.

Parameters

- **session** – the instance of Session used for APIC communication
- **fault_filter** – fault_filter is used to filter the attributes of a fault. given in a hash format with domain, types, severity
- **tenant_name** – tenant_name is a string

get_faults_by_filter(*fault_filter=None*)
filters a fault obj based on the keys given in fault_filter

Parameters **fault_filter** – fault_filter is used to filter the attributes of a fault. given in a hash format with domain, types, severity

Returns fault obj if it satisfies fault_filter

classmethod has_faults(*session*, *fault_filter=None*)
Check for pending events from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **fault_filter** – fault_filter is used to filter the attributes of a fault. given in a hash format with domain, types, severity

Returns True or False. True if there are events pending.

is_deleted()
Not supported

Raises AttributeError

mark_as_deleted()
Not supported

Raises AttributeError

classmethod subscribe_faults(*session*, *fault_filter=None*, *only_new=False*)
Subscribe to faults from the APIC that pertain to instances of this class.

Parameters

- **session** – the instance of Session used for APIC communication
- **fault_filter** – fault_filter is used to filter the attributes of a fault. given in a hash format with domain, types, severity
- **only_new** – Boolean indicating whether to get all events or only the new events. All events (indicated by setting only_new to False) will queue a create event for all of the currently existing objects. Setting only_new to True will only queue events that occur after the initial subscribe. The default has only_new set to False.

classmethod validate_fault_filter(*fault_filter=None*)
validates the fault_filter with the schema

Parameters **fault_filter** – fault_filter is used to filter the attributes of a fault. given in a hash format with domain, types, severity

CHAPTER 7

Applications

ACI Endpoint Tracker

The ACI Endpoint Tracker application tracks all of the attachment, detachment, and movement of Endpoints on the ACI fabric. It stores all of this activity in a database that allows administrators to examine and query the data to gain deep visibility into what is happening in the network. The database also provides a foundation for visualization and querying tools.

Some sample questions that can be answered with the ACI Endpoint Tracker:

- What are all of the current Endpoints on the network ?
- Where is a specific Endpoint ?
- What was connected to the network last Thursday between 3:30 and 4:00 ?
- What are all of the Endpoints belonging to a given Tenant ?
- What Endpoints are on this subnet ?
- What is the history of a given Endpoint (i.e. movement, etc.)?

Installation

acitoolkit

This application uses the acitoolkit. The installation steps for the acitoolkit can be found at http://datacenter.github.io/acitoolkit/.

MySQL database

The ACI Endpoint Tracker uses the open source MySQL database to store the Endpoint data. MySQL is installed separately and the installation steps are dependent on the platform. It is recommended that MySQL be installed in the same machine as the ACI Endpoint Tracker.

MySQL installation instructions for most platforms can be found here: MySQL installation instructions.

For Ubuntu, the installation instructions can be found here: Ubuntu MySQL installation instructions.

Once the above package is installed, you should verify that the MySQL database is running. In Linux and Mac OS X, this can be done by entering the following command:

```
mysqladmin -u root -p status
```

If the database is running, the output should be similar to below:

```
Uptime: 358118  Threads: 3  Questions: 5767  Slow queries: 0
Opens: 109  Flush tables: 1  Open tables: 61  Queries per second
avg: 0.016
```

If the database is not running, the output should be similar to below:

```
mysqladmin: connect to server at 'localhost' failed
```

MySQL Connector

In order for the ACI Endpoint Tracker to communicate with the MySQL database, the MySQL Connector/Python must be installed. This is available for most platforms at http://dev.mysql.com/downloads/connector/python/.

Flask

Flask is required for the optional GUI frontend. The installation steps for Flask can be found at http://flask.pocoo.org/.

Usage

The ACI Endpoint Tracker serves as a conduit between the APIC and the MySQL database. It requires login credentials to both, namely the username, password, and IP address or URL.

The user can choose any **one** of 3 ways to specify the login credentials. If multiple ways are used, they are taken in the following priority order:

1. **Command Line Arguments**

   The login credentials can be passed directly as command line arguments. The command is shown below:

   ```
   python aci-endpoint-tracker.py [[ -u | --url] <apicurl>]
   [[ -l | --login ] <apicusername>] [[ -p | --password ]
   <apicpassword>] [[ -i | --mysqlip ] <mysqlip> [[ -a
   | --mysqladminlogin ] <mysqladminlogin] [[ -s | --mysqlpassword ]
   <mysqlpassword> ]
   ```

   where the parameters are as follows:

apicurl	The URL used to communicate with the APIC.
~~apicusername~~	~~The username used to login to the APIC.~~
apicpassword	The password used to login to the APIC.
~~mysqlip~~	~~The IP address of the MySQL DB host.~~
mysqladminlogin	The username used to login to the MySQL DB
~~mysqlpassword~~	~~The password used to login to the MySQL DB~~

An example would be the following:

```
python aci-endpoint-tracker.py -u https://172.35.200.100 -l
admin -p apicpassword -i 127.0.0.1 -a root -s mysqlpassword
```

2. **Environment Variables**

   The login credentials can be pulled from environment variables in operating systems such as Mac OS X and various Linux distributions.

   The environmental variables are as follows:

```
APIC_URL
APIC_LOGIN
APIC_PASSWORD
APIC_MYSQLIP
APIC_MYSQLLOGIN
APIC_MYSQLPASSWORD
```

   These variables should be set to the correct value. Setting the environment variable is OS dependent. For example, in Mac OS X, environment variables can be set in your `~/.bash_profile` as follows:

```
export APIC_URL=https://172.35.200.100
export APIC_LOGIN=admin
export APIC_PASSWORD=apicpassword
export APIC_MYSQLIP=127.0.0.1
export APIC_MYSQLLOGIN=root
export APIC_MYSQLPASSWORD=mysqlpassword
```

   If environmental variables are used to specify the credentials, then the following command will execute the ACI Endpoint Tracker.:

```
python aci-endpoint-tracker.py
```

3. **Importing a credentials.py file**

   Alternatively, the login credentials can be pulled from a python file named `credentials.py`. In this file, it is assumed that the following variables will be set appropriately for your environment.:

```
URL = 'https://172.35.200.100'
LOGIN = 'admin'
PASSWORD = 'apicpassword'
MYSQLIP = '127.0.0.1'
MYSQLLOGIN = 'root'
MYSQLPASSWORD = 'mysqlpassword'
```

If a credentials.py file is used to specify the credentials, then the following command will execute the ACI Endpoint Tracker.:

```
python aci-endpoint-tracker.py
```

What's it doing ?

Once the ACI Endpoint Tracker is running, it will connect to the APIC and pull all of the existing static and dynamic endpoints that are currently connected to the fabric along with the relevant associated information such as:

- Tenant, Application Profile, and EPG membership
- Interface to which it is connected
- Timestamp of when it connected to the fabric

This data is then inserted into a database called `acitoolkit` that the ACI Endpoint Tracker will create. Within the database, it creates a single table called `endpoints` where all of the endpoint information will be inserted.

Once all of this information is collected, the ACI Endpoint Tracker subscribes through the web socket interface to any updates to both static and dynamic endpoints. When these updates such as endpoint attachment, detachment, or move occurs, the database will be immediately updated with the live data.

Note that updates to the database will only occur when the ACI Endpoint Tracker is running.

Direct Database Query

Once the data is in the database, the MySQL client can be used to query the data directly. Using this method, the full power of SQL can be used to provide deep insight into the network endpoint behavior.

To connect to the MySQL database, you can execute the following command locally on the same host where the database is running.:

```
mysql -u <mysqllogin> -p
```

The client will then prompt for the MySQL database password. After successfully entering the password, the MySQL prompt will come up as shown in the screenshot below:

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 145
Server version: 5.6.22 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

At this point, the `acitoolkit` database should be available. The available databases can be shown by entering the following command at the prompt.:

```
mysql> show databases;
```

A sample output is shown below.:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| acitoolkit         |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

To switch to the `acitoolkit` database, enter the following command.:

```
mysql> use acitoolkit;
```

The endpoint data is stored in a single table called `endpoints`. You can then display all of the endpoint data by the following query (shown with a snippet of the output).:

```
mysql> select * from endpoints;
+-------------------+---------------+--------------+--------------+--------------+-----
↪-----------+---------------------+---------------------+
| mac               | ip            | tenant       | app          | epg          |␣
↪interface      | timestart           | timestop            |
+-------------------+---------------+--------------+--------------+--------------+-----
↪-----------+---------------------+---------------------+
| 74:26:AC:76:80:5B | 192.168.1.133 | Tenant1      | Application1 | WEB          |␣
↪VPC1           | 2014-12-09 19:08:27 | 0000-00-00 00:00:00 |
| 00:50:56:94:D8:73 | 0.0.0.0       | Tenant1      | Application1 | WEB          | eth␣
↪1/102/1/12 | 2015-01-13 23:48:15 | 0000-00-00 00:00:00 |
| 00:50:56:94:07:7E | 0.0.0.0       | Tenant1      | Application1 | WEB          | eth␣
↪1/103/1/11 | 2014-12-19 00:58:16 | 0000-00-00 00:00:00 |
| 00:50:56:94:9A:1C | 192.168.0.5   | Tenant5      | Application1 | USER         | eth␣
↪1/102/1/12 | 2015-01-05 15:29:13 | 0000-00-00 00:00:00 |
| 00:50:56:94:F3:CD | 0.0.0.0       | Tenant5      | Application1 | USER         | eth␣
↪1/102/1/12 | 2015-01-13 23:49:33 | 0000-00-00 00:00:00 |
| 00:50:56:94:17:5E | 0.0.0.0       | Tenant5      | Application1 | WEB          | eth␣
↪1/102/1/12 | 2015-01-10 01:55:40 | 0000-00-00 00:00:00 |
| 00:50:56:94:A9:B5 | 10.0.0.5      | Tenant5      | Application1 | WEB          | eth␣
↪1/102/1/12 | 2015-01-05 15:29:13 | 0000-00-00 00:00:00 |
| 00:50:56:94:93:6F | 0.0.0.0       | Tenant5      | Application1 | WEB          | eth␣
↪1/102/1/12 | 2015-01-10 01:55:40 | 0000-00-00 00:00:00 |
```

At this point, we can query the data using the SQL SELECT command. If you haven't used SQL before, you may want to spend some time learning some of the basic syntax related to the SQL SELECT command as it forms the basis for all queries in the database.

Here are just a few example queries that are possible.

Various fields can be used to filter the results.

Show all of the endpoint information for a specific tenant:

```
mysql> select * from endpoints where tenant='cisco';
```

Show all of the endpoints for a given EPG within a certain tenant:

```
mysql> select * from endpoints where tenant='cisco' and epg='WEB';
```

Show all of the endpoints that were on the network on 2014-12-25:

```
mysql> select * from endpoints where timestart <= '2014-12-25'
and timestop > '2014-12-24';
```

Show all of the history (attach, detach, move) for a particular endpoint:

```
mysql> select * from endpoints where ip='10.1.1.1' and
tenant='cisco';
```

Output can be limited to certain fields

Show the current location of a given endpoint:

```
mysql> select interface from endpoints where ip='10.1.1.1' and
tenant='cisco';
```

Unique fields can be shown using the `distinct` keyword.

Show the EPGs with active endpoints on 2014-12-25:

```
mysql> select distinct tenant,app,epg from endpoints where
timestart <= '2014-12-25' and timestop > '2014-12-24';
```

Counts can be provided for filtered data using the `count` keyword.

Show the number of Endpoints active on 2014-12-25:

```
mysql> select count(*) from endpoints where timestart <=
'2014-12-25' and timestop > '2014-12-24';
```

Wildcarding can be used with the `%` wildcard.

Show the endpoints belonging to a given subnet:

```
mysql> select * from endpoints where ip like '10.10.%';
```

GUI FrontEnd

In addition to the very powerful MySQL interface, there is also a GUI frontend that allows quick simple searching on the database using a web browser. The GUI frontend leverages the DataTables package.

Demo

The usage of this GUI should be fairly intuitive and a live demo with fake endpoint data can be found at the link below. Please give it a try, specifically the Search function to get a feel for how it works.

ACI Endpoint Tracker GUI Demo

For instance, to see all of the endpoints for tenant 'cisco' simply type cisco in the Search box. To narrow the search further to the endpoints owned by tenant 'cisco' on leaf 102, type 'cisco 102' in the Search box. Also, each column can be sorted by clicking on the arrows found in each of the column headers.

Usage

To use the GUI front end locally on your own database, you simply need to execute the `aci-endpoint-tracker-gui.py` file assuming you have installed the Flask package as mentioned in the *Installation* section.

The GUI front end deals exclusively with the MySQL database and does not communicate with the APIC, so it only requires the MySQL credentials. These can be passed in the same manner as described for the ACI Endpoint Tracker *credentials*.

```
python aci-endpoint-tracker-gui.py -i 127.0.0.1 -a root -s
mysqlpassword
```

It should be noted that while the GUI does not communicate with the APIC, as long as the Endpoint Tracker is running, the database will contain the live data for the APIC.

License

ACI Lint

`acilint` is a static analysis tool for Cisco ACI fabrics. Some example use cases for such a tool include the following:

- Configuration Analysis

  In this purpose, it can be used to examine the APIC configuration and determine whether any of the configuration could be possibly problematic or suspicious. It examines the configuration much like static code analysis tools such as the original *lint* checker did for software development in *C* or *pylint* for *Python*. It generates **Warnings** and **Errors** that give indications that the configuration should be examined. Often these Warnings are not problems, but incomplete or stale configuration that is not currently in use.

- Compliance, Governance, and Auditing

  In this purpose, it can be used to determine whether the configuration meets higher level goverance and compliance rules. These rules are similar to *lint* style rules but exploit the APIC ability to provide additional classification tags on objects. Tags provide a simple and flexible way to classify any APIC object in one or more user-defined groups.

For example, EPGs can be tagged as secure and non-secure. A compliance rule can be defined that specifies that secure EPGs cannot consume a contract from a non-secure EPG. Upon violation of this rule, a **warning** or an **error** can be raised.

Usage

`acilint` can be run against the current running APIC configuration or a previously saved set of configuration snapshot files.

Running using Live APIC configuration

When `acilint` collects the configuration directly from the APIC, it needs the proper login credentials. These can be passed via the command line arguments, a `credentials.py` file, environment variables, or if none of these, the user will be directly queried.

The following example shows how to run using the command line arguments for credentials:

```
python acilint.py -l admin -p password -u https://1.2.3.4
```

where `admin` is the APIC login username, `password` is the APIC password, and `https://1.2.3.4` is the URL used to login to the APIC.

Running using Configuration Snapshot files

The `snapback` application provides the ability to save snapshots of the APIC configuration into JSON files. `acilint` can use these snapshot files as input rather than connecting to a live APIC.

This can be useful in debugging as it lets the user compare the `acilint` output of the live APIC to the output of a previous configuration snapshot. `acilint` can also be used to perform some "What If" scenarios. A single configuration snapshot actually consists of multiple snapshot files. These configuration files are then fed as input into `acilint`. These snapshot files fed into `acilint` can actually be from different configuration snapshot versions creating an entirely new configuration that may have never existed on the APIC, but we can run `acilint` against this configuration to check for possible errors and warnings that would occur if this configuration were to be deployed.

The following example shows how to run with configuration snapahot files as input:

```
python acilint.py --snapshotfiles infra.json tenant-cisco.json fabric.json
```

Customization

By default, all checks will be performed. However, like many static code analysis tools, `acilint` is customizable and only the desired warnings and errors can be issued.

To customize `acilint`, generate a configuration file with the following command:

```
python acilint.py --generateconfigfile acilint.cfg
```

or even shorter:

```
python acilint.py -g acilint.cfg
```

where `acilint.cfg` is the filename you wish to create.

The generated configuration file will contain a list of all of the current checks being performed.

An example config file is shown below:

```
# acilint configuration file
# Remove or comment out any warnings or errors that you no longer wish to see
 error_001
 error_002
 warning_001
 warning_002
 warning_003
```

To remove checks, either:

- Delete the line containing the check, or
- Comment it out by prepending a # in front of the check

Errors and Warnings

The following list of Errors and Warnings are performed by `acilint`. Since `acilint` is written on top of the `acitoolkit` package, the checks are limited to the functionality exposed by that package. However as the `acitoolkit` expands, so shall `acilint`.

Warnings

warning_001	Tenant has no app profile
warning_002	Tenant has no context
warning_003	AppProfile has no EPGs
warning_004	Context has no BridgeDomain
warning_005	BridgeDomain has no EPGs assigned
warning_006	Contract is not provided at all
warning_007	Contract is not consumed at all
warning_008	EPG providing contracts but in a Context with no enforcement
warning_010	EPG providing contract but consuming EPG is in a different context
warning_011	Contract contains bi-directional TCP Subjects
warning_012	Contract contains bi-directional UDP Subjects
warning_013	Contract has no Subjects
warning_014	Contract has Subjects with no Filters

Errors

error_001	BridgeDomain has no context
error_002	EPG has no BD assigned
error_005	Duplicate or overlapping subnets in Context
error_006	ExternalNetwork Subnets duplicated in fabric

Critical

critical_001	Compliance check example

critical_001 is a compliance check example that will perform the following:

- Ensure that all of the EPGs in the system have been classified as *secure* and *nonsecure* using the tagging capability provided by the `acitoolkit`.
- Ensures that none of the *secure* EPGs can communicate with the *nonsecure* EPGs by checking that no contract provided by *secure* EPGs is consumed by *nonsecure* EPGs.

Developing Checks

Additional checks can be added through new methods on the `Checker` class. If the method begins with `warning_` or `error_` or `critical_`, it will automatically be executed as part of the `acilint` execution. The new checks will also automatically inherit the customization capability through the usage of the configuration file. Some familiarity with the `acitoolkit` object model is necessary to write additional checks.

Cableplan Application

The Cable Plan module allows the programmer to easily import existing cable plans from XML files, import the currently running cable plan from an APIC controller, export previously imported cable plans to a file, and compare cable plans.

More advanced users can use the Cable Plan to easily build a cable plan XML file, query a cable plan, and modify a cable plan.

Using the Cable Plan

Invoking

The Cable Plan module is imported from `cableplan.py` which can be found in the `acitoolkit/applications/cableplan` directory.

It can be incorporated directly into a python script, or it can be used from the command-line.

```
>>>from cableplan import CABLEPLAN
```

When you want to create a cable plan from the current running topology of an ACI fabric, simply do the following:

```
>>>cp = CABLEPLAN.get(session)
```

Where `session` is an ACI session object generated using the `acitoolkit`. `cp` will be the cable plan object.

You can export that cable plan by opening a file and calling the `export()` method as follows:

```
>>>cpFile = open('cableplan1.xml','w')
>>>cp.export(cpFile)
>>>cpFile.close()
```

The cable plan will be written to the `cableplan1.xml` file.

Reading an existing cable plan xml file is equally easy.:

```
>>>fileName = 'cableplan2.xml'
>>>cp2 = CABLEPLAN.get(fileName)
```

Note that you don't have to explicitly open or close the file. The `get(fileName)` method will take care of that for you.

Comparing cable plans is one of the more interesting cabablilities of the Cable Plan module and is very easy to do using the "difference" methods. When generating the difference between two cable plans, the module will return those items that exist in the first cable plan, but not in the second.

For example, assume that in the above example, the second cable plan read from the `cableplan2.xml` file does not have switch "Spine3" and the first cable plan does have it. The following example will print all of the switches in the first cable plan and not in the second.:

```
>>>missing_switches = cp1.difference_switch(cp2)
>>>for switch in missing_switches :
>>>    print switch.get_name()
Spine3
```

Similiarly, the following example will print all of the missing links:

```
>>>missing_links = cp1.difference_link(cp2)
>>>for link in missing_links :
>>>    print link.get_name()
```

To understand all of the differences between two cable plans it is necessary to compare them in both directions

```
>>>missing_links = cp1.difference_link(cp2)
>>>extra_links = cp2.difference_link(cp1)
>>>print 'The following links are missing from the second cable plan'
>>>for link in missing_links :
>>>    print link.get_name()
>>>print 'The following links are extra links in the second cable plan'
>>>for link in extra_links:
>>>    print link.get_name()
```

If multiple ports are specified in the link object with minPorts and maxPorts attributes (see Cable Plan XML Syntax below), it is possible that a link object in the first cable plan is only partially met by the link objects in the second cable plan. The `remaining_need()` method of the CpLink object.:

```
>>>missing_links = cp1.difference_link(cp2)
>>>for link in missing_links :
>>>   print 'Link',link.get_name(), 'still needs',link.remaining_need(),'links to␣
↪satisfy its mimimum requirement'
```

There is a similar method,

>>>missing_links = cp1.difference_link(cp2) >>>for link in missing_links : >>> print 'Link',link.get_name(), 'still needs',link.remaining_need(),'links to satisfy its mimimum requirement'

There is a similar method,

>>>missing_links = cp1.difference_link(cp2) >>>for link in missing_links : >>> print 'Link',link.get_name(), 'still needs',link.remaining_need(),'links to satisfy its mimimum requirement'

There is a similar method, `remaining_avail()` that returns the number of physical links the link object could match.

The `remaining_need()` and `remaining_avail()` values are reset when the `difference_link()` method is invoked.

It might be necessary to compare cable plans when the names of the switches are different, but the topologies are the same. This can easily done by simply changing the names of the switches that are different and then doing the comparisons.:

```
>>>switch = cp1.get_switch('Spine1')
>>>switch.set_name('Spine1_new_name')
```

This will automatically also fix-up all of the link names that are connected to the switch whose name is being changed. Note that this is also an easy way to change the name of a switch in a cable plan file. Simply read it in, change the switch name, and export it out. The following example will read in `cable_plan2.xml`, change the name of 'Leaf1' to 'Leaf35', and then export to the same file the modified cable plan:

```
>>>fileName = 'cable_plan2.xml'
>>>cp2 = CABLEPLAN.get(fileName)
>>>switch = cp2.get_switch('Leaf1')
>>>switch.set_name('Leaf35')
>>>f = open(fileName,'w')
>>>cp2.export(f)
>>>f.close()
```

Cable Plan from the Command Line

Invoking the cable plan application from the command line is very simple.

From the command prompt do the following:: > python cableplan.py -h

This will then return usage instructions that explain each of the command line options.

There are two primary functions that can be invoked from the command-line: 'export' and 'compare'.

The 'export' function, selected with the '-e' option, will create a cable plan by reading the state of the ACI fabric from the APIC controller. This cable plan will be either displayed on the monitor or, if a filename is specified, will be placed in a file. It will be in nicely formatted XML.

The 'compare' function will compare two cable plans. One of those must be in a file that is specified at the command line and the second one can come either directly from the APIC or from a second file. If only the '-c1 file_name' option is used, then the content of file_name is compared to the actual running configuration of the ACI fabric.:

```
> python cableplan.py -c1 netwrk1.xml
```

If you want to compare two files, then both the '-c1 file_name1' and '-c2 file_name2' options must be used.:

```
> python cableplan.py -c1 netwrk1.xml -c2 netwrk2.xml
```

This comparison will list all of the links in the first cable plan that are not in the second and vice-versa.

Cable Plan XML Syntax

The cable plan XML looks like the following

```
<?xml version="1.0" encoding="UTF-8"?>
<?created by cable.py?>
<CISCO_NETWORK_TYPES version="None" xmlns="http://www.cisco.com/cableplan/Schema2"␣
↪xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="nxos-
↪cable-plan-schema.xsd">
   <DATA_CENTER networkLocation="None" idFormat="hostname">
      <CHASSIS_INFO sourceChassis="spine1" type="n9k">
         <LINK_INFO sourcePort="eth2/35" destChassis="leaf1" destPort="eth1/50"/>
         <LINK_INFO sourcePort="eth2/3" destChassis="leaf3" destPort="eth1/50"/>
         <LINK_INFO sourcePort="eth2/2" destChassis="leaf2" destPort="eth1/50"/>
```

```
      </CHASSIS_INFO>
      <CHASSIS_INFO sourceChassis="spine2" type="n9k">
         <LINK_INFO sourcePort="eth2/1" destChassis="leaf1" destPort="eth1/49"/>
         <LINK_INFO sourcePort="eth2/3" destChassis="leaf3" destPort="eth1/49"/>
         <LINK_INFO sourcePort="eth2/2" destChassis="leaf2" destPort="eth1/49"/>
      </CHASSIS_INFO>
   </DATA_CENTER>
</CISCO_NETWORK_TYPES>
```

The CHASSIS_INFO tag normally identifies the spine switches and the leaf switches are contained in the LINK_INFO. When the XML is read in, both leaf and spine switch objects will be created and the `get_switch()` and `get_link()` methods can be used to access them.

The LINK_INFO syntax also allows more flexible and loose specifications of the links. If the `sourcePort` or `destPort` attributes are left out, then any port on that corresponding switch can be used. The `sourcePort` and `destPort` attributes can also take port ranges, and lists as shown here:

```
<LINK_INFO sourcePort="eth1/1-eth1/15, eth1/20" destChassis =
"leaf3"/>
```

In addition, you can add `minPorts` and `maxPorts` attributes to specify the minimum number of ports or maximum number of ports when multiple are defined.:

```
<LINK_INFO sourcePort="eth2/3, eth3/4 - eth3/10",
destChassis="leaf2", destPort="eth1/1 - eth1/8", minPorts=3,
maxPorts=5>
```

If minPorts is omitted, the default will be 1. If maxPorts is omitted, the default will be unlimited.

When comparing two cable plans using the `difference_link()` method, if the minimum number of links in the first cable plan can be met with second cable plan, then the difference will show no difference. Note that it is possible that requirements of several links specified in one cable plan may be met by one or more links in the other. Basically, the difference is calculated such that the minimum requirements of the first cable plan are met without exceeding the maximum capacity of the second cable plan.

Cable Plan API Reference

class cableplan.**CpPort**(*port_set*)

This class holds the information for a link's port. Since the port can be a single port, a list or a range, putting it in a class allows more flexible operations on it.

list()

name()

remove_available_port(*port*)

reset_accounting()

class cableplan.**CpLink**(*source_chassis*, *dest_chassis*, *source_port=None*, *dest_port=None*, *min_ports=None*, *max_ports=None*)

export(*chassis*, *level*)

Will return string of XML describing the LINK_INFO. It will use 'chassis' to determine which is the source chassis so that it will be omitted from the XML and the other chassis will become the destination. 'level' is the indentation level.

Parameters

- **chassis** – Chassis that is the parent of the LINK_INFO xml
- **level** – Indentation level

Returns str

get_name()

has_port_in_common(*link*)
Returns True if link has any ports that match self. It will compare all ports included expanded lists of port sets.

Parameters **link** – link to check to see if matches, or overlaps, with self

Returns Boolean

is_connected(*switch1*, *switch2=None*)
Returns True if switch1 is one of the switch endpoints of the link and switch2 is unspecified otherwise is will return True if both switch1 and switch2 are switch endpoints of the link. If switch1 is the same as switch2, it will return False.

Parameters

- **switch1** – first switch to check if it an end-point of the link
- **switch2** – optional second switch to check if it an end-point of the link

Returns True if switch1 (and optional switch2) is an end-point of the link

static **match_links**(*link1*, *link2*)
This will match-up link1 and link2 and increment the reference count in each link for each of the matches that happen. It will do this until the minimum number of links has been reached for link1. It will return the number of matches that occurred.

Parameters

- **link1** – first link of type CpLink that is part of the matching
- **link2** – second link of type CpLink that is part of the matching

Returns number of matches that occured.

order()
Calculates the order of the link defined by the maximum number of physical links this link can represent

Returns int

remaining_avail()
returns the remaining number of physical links available to match against The parameters used to calculate this value are reset by the reset_accounting() method which is typically invoked when invoking a difference_link() method on the CABLEPLAN parent object.

Returns int

remaining_need()
returns the remaining number of physical links needed to match against self to satisfy requirements. The parameters used to calculate this value are reset by the reset_accounting() method which is typically invoked when invoking a difference_link() method on the CABLEPLAN parent object.

Returns int

reset_accounting()
Resets account on the source and dest ports as well as reference count

class cableplan.**CpSwitch**(*name*, *chassis_type=None*, *spine=False*, *parent=None*)
Bases: object

class holding a switch

export(*level*)

get_links()
returns a list of CP_LINKS from the parent CABLEPLAN that are connected to self.

Returns list of CP_LINKS

get_name()
Gets the name of the chassis.

Returns str

get_type()
Gets the chassis type. Examples of chassis types are 'n7k' or 'n9k'

Returns str

is_spine()
Checks if the 'spine' flag is set.

Returns True if the spine flag is set, otherwise False

merge(*new_switch*)
Merges the content of new_switch with self. If self has variables set, then they will not be changed. If they have not been set, then they will be assigned the value from new_switch.

Parameters **new_switch** – switch object to merge with self

set_name(*name*)
Sets the switch name. This will over-ride any preexisting name. Note that this new name will now become part of the link name for all the links attached to this switch.

Parameters **name** – name string to set in the switch

set_parent(*parent*)
Sets the parent of the switch. Parent must be of type CABLEPLAN. If a parent CABLEPLAN was already set and it is differnt from parent, then an error is raised.

Parameters **parent** – parent object of type CABLEPLAN

class cableplan.**CABLEPLAN**(*version=None*)

add_link(*new_link*)
Will add a link to the CABLEPLAN. Duplicates will not be allow, but overlapping will be.

Parameters **new_link** – Link to be added of type CpLink

Returns None

add_switch(*new_switch*)
This will new_switch to the CABLEPLAN. If the switch already exists, it will merge the new_switch with the existing one. It will also set the parent of the switch to be the CABLEPLAN. It will return the final switch, i.e. new_switch if no merge occurred or the newly merged switch if a merge did occur.

Parameters **new_switch** – switch to be added of type CpSwitch

Returns CpSwitch

delete_link(*link*)

delete_switch(*old_switch*)

difference_link(*cp*)
returns a list of links that are in self, but not in cp.

Parameters **cp** – cable plan of type CABLEPLAN

Returns list of CpLink

difference_switch(*cp*)
Will return a list of switches that are in self, but not in cp.

Parameters **cp** – cable plan

Returns list of CpSwitch

exists_link(*link*)

exists_switch(*switch*)

export(*out_file=None*, *level=0*)
Will generate XML text of the entire CABLEPLAN and return it as a string. If out_file is specified, it will write the XML to that file. out_file should be opened for writing before calling this method. 'level' specifies the amount of indentation to start with.

export_data_center(*level=0*)
Will generate the XML of the CABLEPLAN with DATA_CENTER as the root. This will then be returned a string. 'level' specifies the indentation level to start with.

Parameters **level** – optional indention level, integer

Returns string that is the DATA_CENTER xml

classmethod get(*source*)
This will get input a cable plan from 'source'. If source is a string, it will get the cable plan from XML in a file whose name is source. If it is a Session, it will read the corresponding APIC to get the cable plan.

Parameters **source** – filename of type string or Session of type Session

Returns CABLEPLAN

get_links(*switch1=None*, *switch2=None*)
Returns a list of links. If switch is unspecified, it will return all links. If switch is specified, it will return all of the links that are connected to switch. If both switch1 and swithc2 are specified, it will return all links that are connected between the two switches.

Parameters

- **switch1** – optional first switch of type CpSwitch
- **switch2** – optional second switch of type CpSwitch

Returns list of links of type CpLink

get_spines()
Will return list of switches that are spines

Returns list of CpSwitch

get_switch(*switch_name=None*)

reset_accounting()
clears the refernce count on each link :rtype : None

sorted_links(*switch1*, *switch2*)
returns a sorted list of links between switch1 and switch2. They are sorted by specificity from most specific to least specific. The specificity is determined by which list of ports is the minimum between source and destination and which is the minimum across links. :rtype : list :param switch1: :param switch2:

Snapback : Configuration Snapshot and Rollback

Summary

Snapback is a Configuration Snapshot and Rollback tool for ACI fabrics. Specifically, the tool allows an administrator to perform the following tasks:

- Live snapshots of the running ACI fabric configuration
- One-time and recurring snapshots, both immediate and scheduled
- Versioned storage of the configuration
- Full viewing of any snapshot configuration including the differences between snapshots
- Rollback to any previous configuration snapshot; Full or Partial
- Web based or Command Line administration

General Overview

Snapback provides the ability to generate configuration snapshots of the APIC configuration.

Snapshot Files

The configuration snapshots consist of text files containing the system configuration at the time of the snapshot stored in JSON format. Each snapshot is stored in a versioned repository along with a version identifier. The version identifier is automatically generated by concatenating the date with the time of the snapshot.

A single configuration snapshot contains a number of files. All of the configuration under the managed object called fabric is placed in the file `fabric.json`. Similarly, all of the configuration under the managed object called infra is placed in the file `infra.json`. Each tenant has their own configuration file with the file name formatted as `tenant-<tenantname>.json` where `<tenantname>` is replaced with the actual name of the tenant.

Each snapshot will only record the differences between the previous snapshot and the current APIC configuration at the time of the snapshot. This ensures that the minimal amount of state is stored on disk to represent the actual configuration. This also means that even though a snapshot make occur, it may not create a new version of every configuration file. In fact, if there have been no changes at all since the previous snapshot, no new files will be created.

Rollback

Rollback is when a previous configuration snapshot is used to replace the current APIC configuration. This can be done on the granularity of a snapshot file. This means that a single tenant can be moved back to a previous configuration while the other tenants keep their current configuration state. This is often referred to as partial rollback. When all of the configuration files for a particular version are used to replace the current APIC configuration, this is referred to as a full rollback.

Installation

Snapback is part of the acitoolkit and will be installed at the same time. The installation details can be found here.

Web based Usage

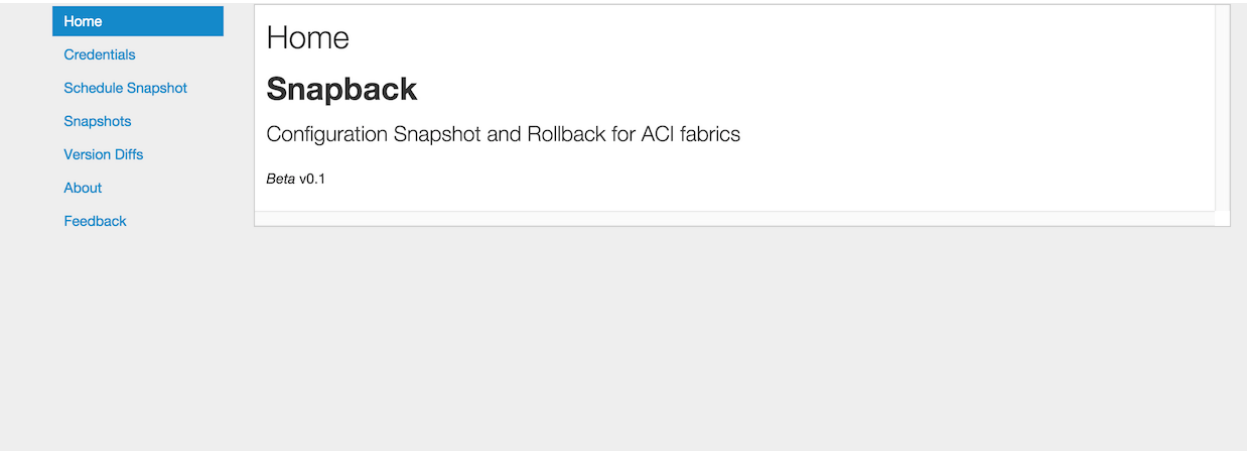
Snapback can be run as a web application. Running Snapback as a web application is done by switching to the snapback directory and running the application:

```
cd acitoolkit/applications/snapback
python snapback.py
```

By default, snapback will run locally on the loopback address. Accessing the tool is done by running a web browser locally and pointing to the following URL:

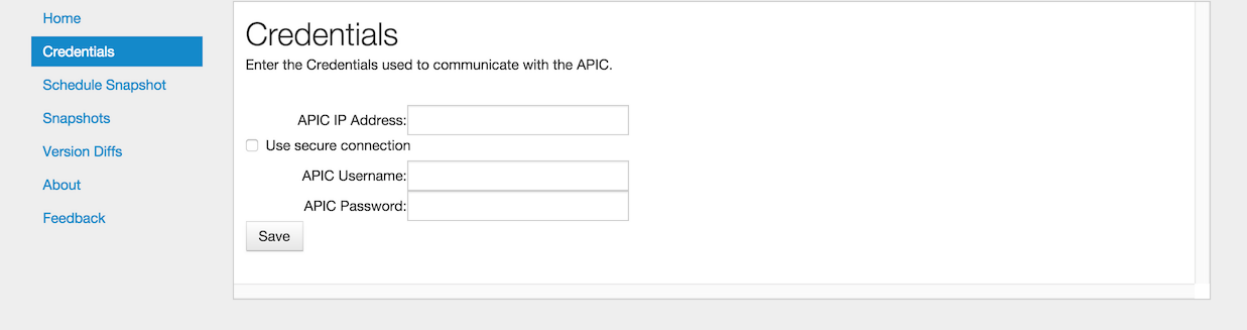
```
http://127.0.0.1:5000/
```

Upon pointing the web browser to the Snapback URL, the Snapback welcome screen along with the navigation menu on the left hand side.



Credentials

Entering the APIC credentials is necessary for Snapback to be able to perform configuration snapshots. Selecting the Credentials in the navigation menu will bring up the Credentials screen where the APIC credentials can be entered.



A number of fields allow the entry of the APIC credentials. These fields are:

APIC IP Address: This field is the IP address used to communicate with the APIC.

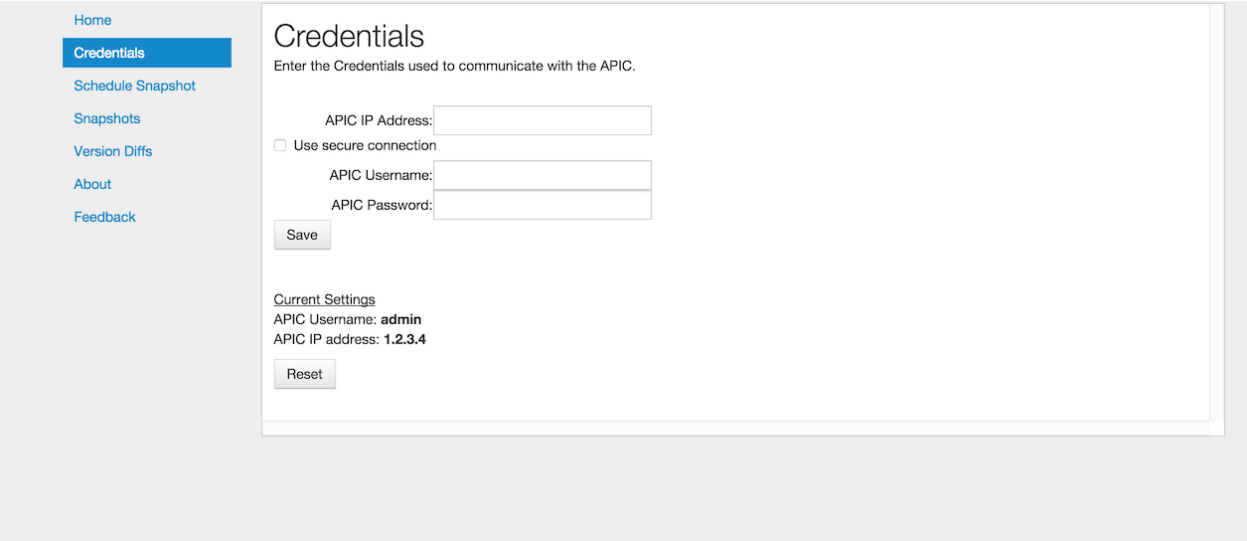
Use secure connection: When selected, this checkbox indicates that the APIC communication uses https instead of http.

APIC Username: The username used when logging in to the APIC.

APIC Password: The password used when logging in to the APIC.

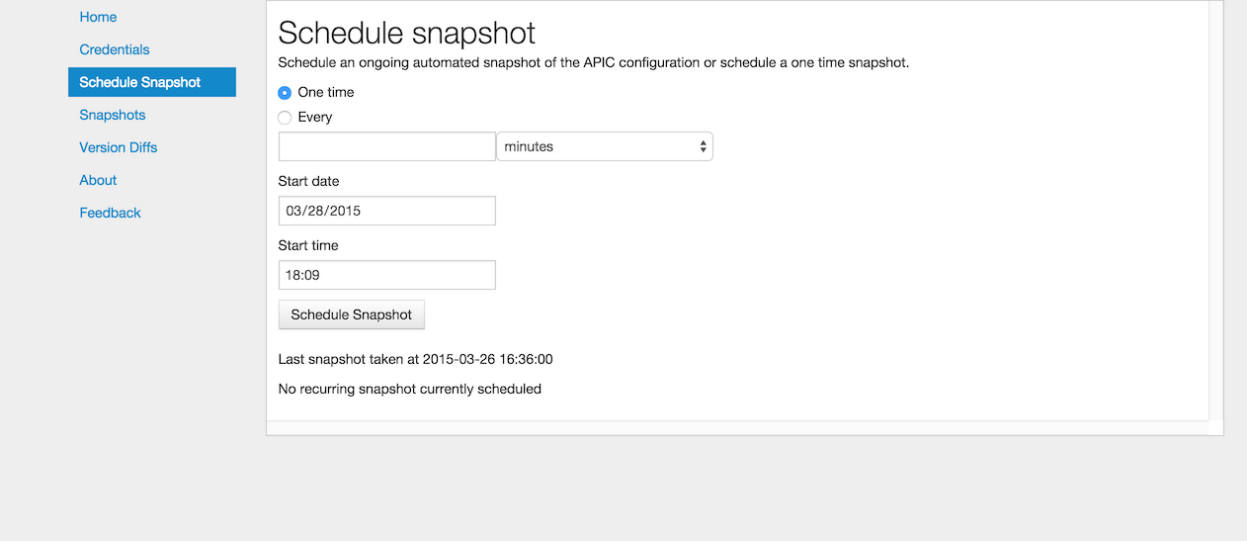
The credentials are stored with Snapback by selecting the Save button. Once the credentials are entered, they will be saved as part of the user session on that browser. This means that another window or tab using Snapback can be opened on the same machine without having to reenter the credentials.

When entered, the current credentials are shown as in the screenshot below. Note that for security purposes, the password is not displayed. Clicking the **Reset** button will cause Snapback to forget the current credentials, but will not impact the APIC in any way.



Schedule Snapshots

Once the credentials have been entered, configuration snapshots can be taken. Snapshots are configured by setting a snapshot schedule. By selecting the Schedule Snapshots in the navigation menu, the following screen will appear.



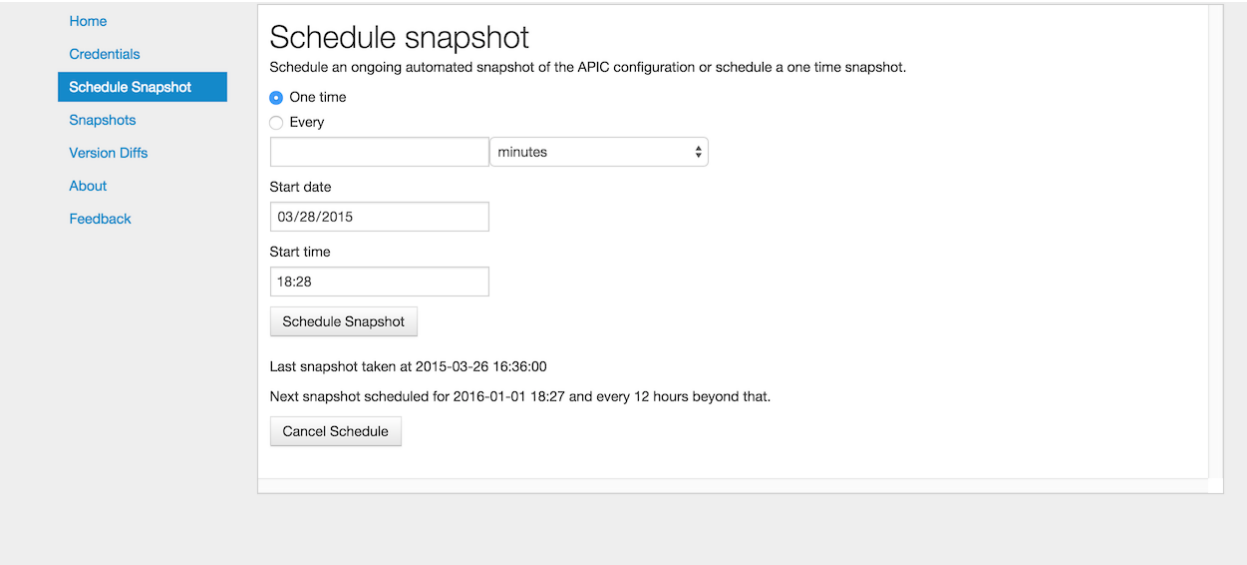
The snapshot can be a one time occurrence or recurring. The *Start date* and *Start time* fields will be used as the date and time that will be used to create the snapshot. If the time has passed, the snapshot will be triggered immediately.

Selecting the radio button labeled *One time* will schedule a single configuration snapshot.

Selecting the radio button labeled *Every* will schedule a recurring configuration snapshot starting at the specified *Start date* and *Start time*. Immediately below the *Every* radio button is the field to enter the time interval for the snapshot. This is entered as a numeric value within the text box and selecting the granularity of the interval from the drop down box. The granularity options are *minutes*, *hours*, or *days*.

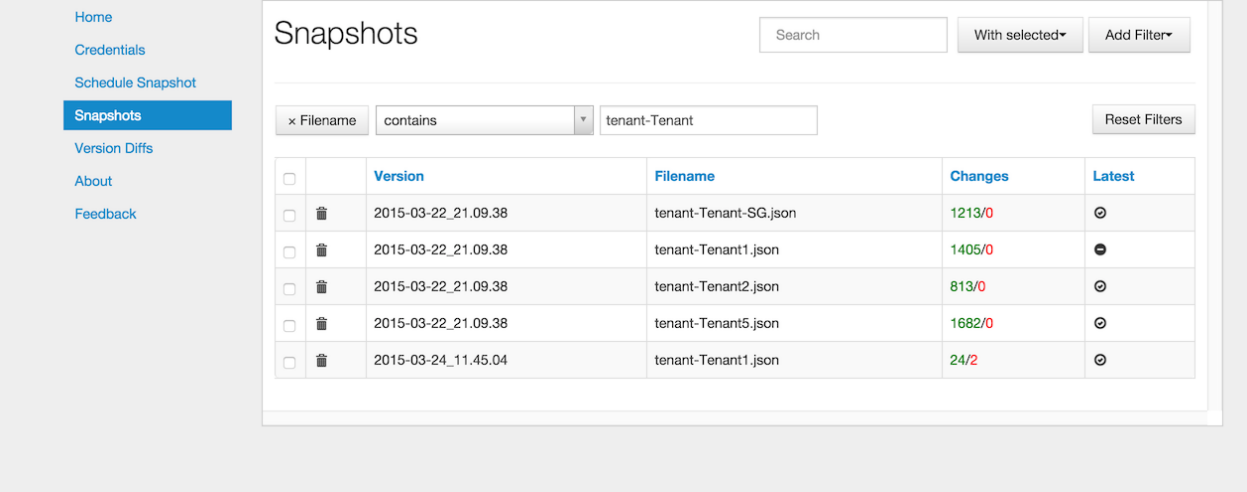
Selecting the *Schedule Snapshot* button will cause the schedule to be submitted to Snapback. Below the *Schedule Snapshot* button, the last successful snapshot time is shown.

Once a configuration snapshot has been scheduled, the option to cancel the current snapshot schedule is shown as in the screenshot below.



Snapshots

Selecting *Snapshots* in the navigation window will bring up the following screen.



The existing snapshots will be shown in a table format. Each row in the table represents a snapshot file. The columns consist of the following:

Version: This is the timestamp of the configuration file. The format is YY-MM-DD_HH:MM:SS

Filename: The name of the configuration file as described in section *Snapshot Files*.

Changes: The Changes column gives the number of lines that have changed in this version as compared to the previous version. The changes are represented as *additions/deletions*. The additions are shown in green text and the deletions are shown in red text.

Latest: This column shows whether the configuration file is the most recent version of configuration. A checkmark indicates that this file is the latest. Since a configuration snapshot file is only created when there are changes in the configuration, the latest version of different configuration filenames may be different.

Each of the columns can be sorted by clicking on the column title.

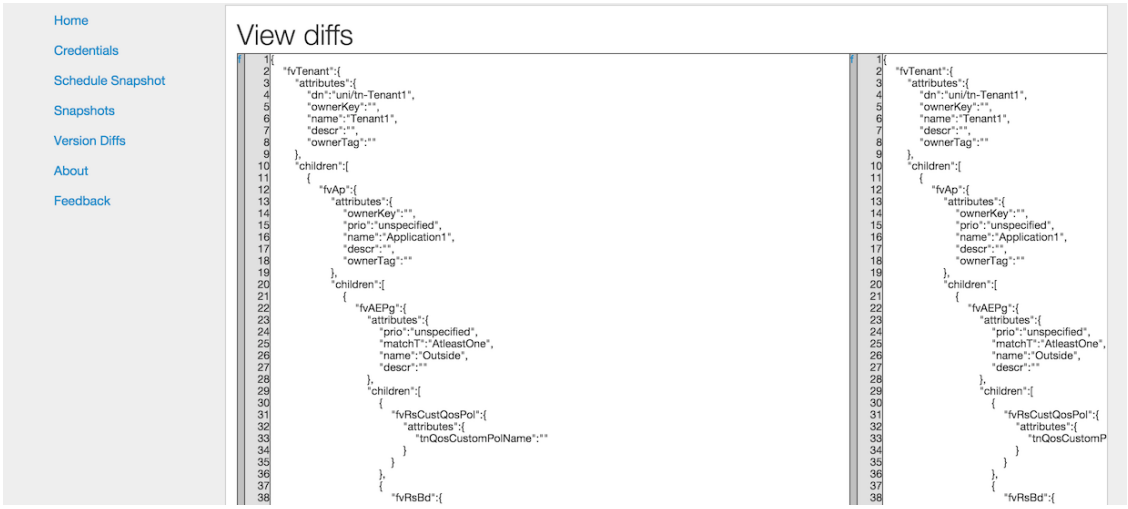
The *Search* text box can be used to filter the table rows shown to the text entered into the box. The filter matching will be performed on the data contained within the Version and Filename columns.

Filters can be added by selecting the *Add Filter* pulldown menu in the top right corner of the screen. Filters can be added for the Version, Filename, and/or Latest columns.

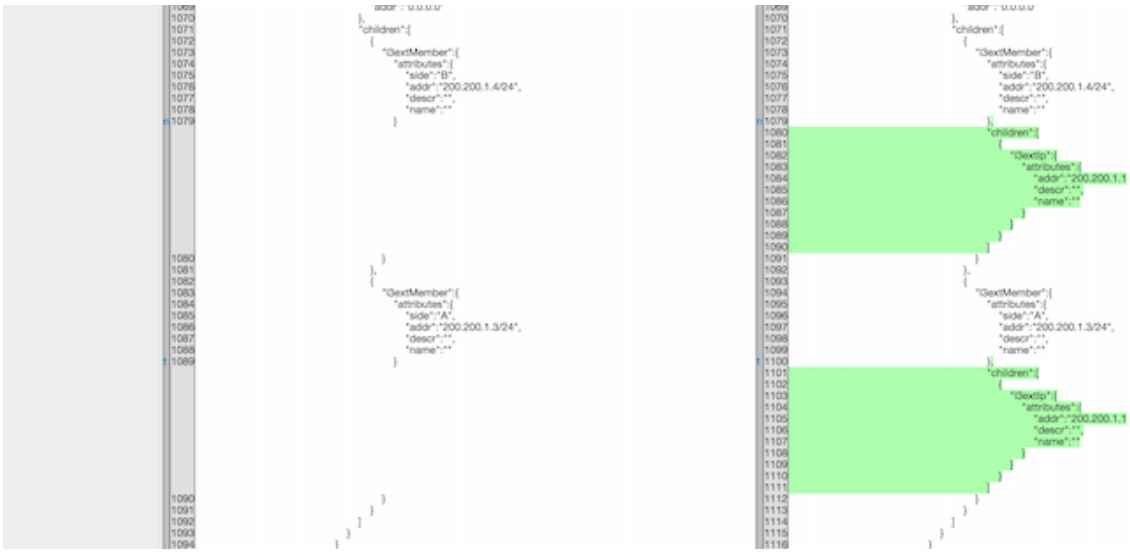
Each row has a checkbox as the leftmost column. Rows can be selected individually or all rows can be selected by checking the checkbox in the column title row. Rows that have been selected can be subject to the actions contained within the *With Selected* pulldown menu. The following options are available within as *With Selected* options:

- **Rollback** Selecting *Rollback* will cause the selected configuration snapshot files to be pushed to the APIC overwriting the current existing configuration. Since this can be a disruptive operation, a confirmation dialog box will appear to confirm the user's intent.
- **View** Selecting *View* will open the selected files in a separate screen showing the entire JSON configuration.

- **View Diffs** Selecting *View Diffs* requires 2 and only 2 files to be selected. In this view, the 2 files are opened in a new screen and shown side-by-side. The 2 files should be of the same file name but different versions.

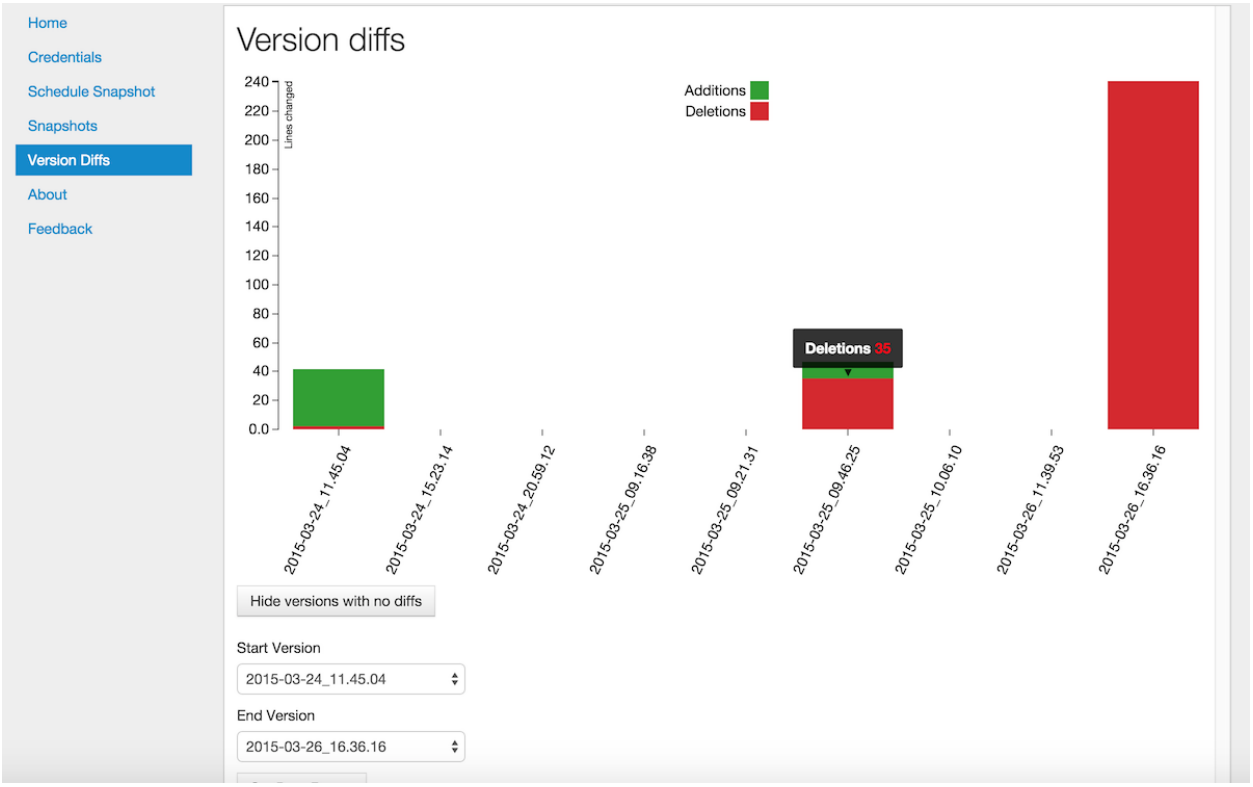


File differences are shown highlighted in green for additions, red for deletions, and yellow for small intra-line changes. The left-most column allows quick navigation between the changes by clicking on the letter within the column. The letter *f* moves to the first change. The letter *n* moves to the next change. The last change will be marked with the letter *t* which when clicked will move to the first change in the file.



Version Diffs

Selecting *Version Diffs* in the navigation will show a visual representation of the differences between the various versions of configuration snapshots.



Within this view, the number of lines changed per version are shown as a stacked bar chart with *Additions* marked as green and *Deletions* marked as red.

Since many configuration snapshots may not contain any changes, there is a button that can be selected to hide or show the versions that have no changes.

Since there may be huge change differences between different configuration versions, it may be useful to narrow the graph to a specific date range. For instance, the very first snapshot will likely have a large number of additions which may visually hide the subsequent changes due to scale. By adjusting the date range, the initial snapshot can be removed from the bar chart.

Hovering the mouse pointer over the bar will show the actual value.

The value is the cumulative changes from all snapshot files within that version. Clicking on the bar will open up the *Snapshots* view with only the files from that configuration version. In the *Snapshots* view, the changes are shown on per configuration file.

About

This is a simple summary screen with a link to the documentation and a description of the software license.

Feedback

This screen provides the ability for the users to submit comments, suggestions, feature requests, and bug reports directly to the authors of the tool.

Command Line Usage

Snapback can also be run as a command line application. This is done by switching to the snapback directory and running the application:

```
cd acitoolkit/applications/snapback
```

The application is run with options provided in the command line. The help for this command is shown by:

```
python aciconfigdb.py --help
```

The output for this command is shown below:

```
usage: aciconfigdb.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD]
                      [-s | -ls | -lc [VERSION [VERSION ...]] | --rollback
                      VERSION [CONFIGFILE ...] | --show VERSION CONFIGFILE]

Configuration Snapshot and Rollback tool for APIC.

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     APIC IP address.
  -l LOGIN, --login LOGIN
                        APIC login ID.
  -p PASSWORD, --password PASSWORD
                        APIC login password.
  -s, --snapshot        Take a snapshot of the APIC configuration
  -ls, --list-snapshots
                        List all of the available snapshots
  -lc [VERSION [VERSION ...]], --list-configfiles [VERSION [VERSION ...]]
                        List all of the available configuration files.
  --rollback VERSION [CONFIGFILE ...]
                        Rollback the configuration to the specified version.
                        Optionally only for certain configuration files.
  --show VERSION CONFIGFILE
```

```
                        Show the contents of a particular configfile from a
                        particular snapshot version.
```

The APIC credentials are provided with the *–url*, *–login*, and *–password* options.

An immediate snapshot is taken by passing the *–snapshot* option.

All of the snapshot versions can be shown by passing the *–list-snapshots* option.

All of the snapshot files can be listed by passing the *–list-configfiles* option. If the optional version is passed, the files from those versions will be listed. If no version is passed, the files from the most recent configuration snapshot will be listed.

Configuration snapshot files can be used to rollback the running APIC configuration by issuing the *–rollback* option. This can be issued for all snapshot files in the version or if the optional list of configuration files is given, the rollback will only occur for the specified files.

The file *aciconfigdb.py* serves a dual purpose. It is both a standalone tool and provides the back end to the GUI. It can be imported as a library and used in your own applications in the same way as used by the GUI.

Version Repository

The core of the configuration snapshot engine is the Git repository. This is automatically created by Snapback and hidden from the user if so desired. The actual repository is created in the directory named *apic-config-db*. All of the snapshot files can be found in this directory and all of the regular git commands can be used on this repository. Each version is stored as a git tag. This allows easy access to any version directly from the command line. If you wish to view the repository directly, the following commands will likely be useful:

```
* git tag

    This will show all of the configuration snapshot versions.

* git show <version>:<filename>

    This will show the contents of the specific version of the
    specified configuration file.

* git diff <version>:<filename> <version>:<filename>

    This can be used to view the differences between 2 versions of
    the same configuration file.
```

Visualization Examples

This directory contains a number of visualization examples that can be used to display information collected using the `acitoolkit`. Many of the examples are meant to run alongside the `ACI Endpoint Tracker` application and interact with the MySQL database that the `ACI Endpoint Tracker` populates. Most of the visualization examples are interactive.

Installation

To run the visualizations, the python package `Flask` is required. This can be installed using `pip` as follows:

```
pip install flask
```

It is also recommended that the `ACI Endpoint Tracker` is installed.

Usage

Run the visualizations as follows (supplying your own MySQL credentials):

```
python acitoolkit-visualizations.py --mysqlip 127.0.0.1 --mysqllogin root --
↪mysqlpassword password
```

Alternatively, you can create a *credentials.py* file in the same directory with the following:

```
MYSQLIP='127.0.0.1'
MYSQLLOGIN='root'
MYSQLPASSWORD='password'
```

If the *credentials.py* file is used, run the visualizations as follows:

```
python acitoolkit-visualizations.py
```

Once the visualizations are running, you should see the following displayed:

```
* Running on http://127.0.0.1:5000/
* Restarting with reloader
```

Simply point your favorite web browser to the following URL and explore:

```
http://127.0.0.1:5000/
```

EventFeeds: ACI Events to Atom Feed

The EventFeeds application subscribes to APIC Managed Objects and records any updates to the objects over a web-socket connection. These updates can be viewed in a variety of Atom Feeds provided over HTTP.

Some sample use cases for the ACI Events to Atom Feed app:

- Display recent endpoints in a NOC
- Display updated tenants on an IPTV
- Monitor EPG changes in a feed client

Installation

acitoolkit

This application uses the acitoolkit. The installation steps for the acitoolkit can be found at http://datacenter.github.io/acitoolkit/.

Flask

Flask is required. Flask should be installed automatically as a dependency of acitoolkit. If not the installation steps for Flask can be found at http://flask.pocoo.org/.

Usage

1. **Command Line Arguments**

   The login credentials can be passed directly as command line arguments. The command is shown below:

```
python eventfeeds.py --help
usage: eventfeeds.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD]
            [--snapshotfiles SNAPSHOTFILES [SNAPSHOTFILES ...]] [--ip IP]
            [--port PORT] [--test]
```

   where the parameters are as follows:

URL	The URL used to communicate with the APIC.
LOGIN	The username used to login to the APIC.
PASSWORD	The password used to login to the APIC.
IP	The IP address the webserver should bind to
PORT	The PORT the webserver should bind to

   An example would be the following:

```
python eventfeeds.py -u https://172.35.200.100 -l
admin -p apicpassword --ip 127.0.0.1 --port 5000
```

2. **Configuration**

   The application can be configured via the GUI at `http://[ip]:[port]/config/`

   Under **Record Events For...** select the classes that you would like the app to monitor.

   Only select the classes that you are interested in to save on network and database usage.

   Log Filename and **Events Database** can be used to set where the log files are saved to and where the sqlite3 event database will be stored. The path is relative to the working directory of the app.

   Clicking **Save Configuration** will write out the configuration to *config.json*

   You **must** restart the process for the changes to take effect.

What's it doing ?

Once the application is running, it will connect to the APIC and subscribes (over a websocket) to any requested classes in the configuration file.

Whenever an update is received a row is inserted into a local database table named `events`. Each row has the Class Name, Name, Timestamp, and JSON representation of the object.

When you request a feed the Flask web application will query the database for the relevant events and then dynamically generate an Atom compatable feed.

Note that updates to the database will only occur when the ACI Events to Atom Feed is running.

Feed Details

The feed produced by this application is compatable with the Atom Syndication Format standard

Each feed entry consists of the following fields:

```
<entry xml:base="http://[ip]:[port]/events/class/[Class]/[Filter]/">
  <title type="text">[Class]</title>
  <id>[Name]</id>
  <updated>[Timestamp]</updated>
  <author>
    <name>APIC</name>
  </author>
  <content type="text">[JSON]</content>
</entry>
```

Note: The JSON body will have the following characters escaped ” ‘ < > &

Screenshots

Head to the root of the application to get a list of available feeds e.g. *http://127.0.0.1:5000/*

ACI Toolkit - Available Feeds

Online Documentation

Configuration

All Subscribed Classes

/events/recent/
/events/recent/day/
/events/recent/week/
/events/recent/month/
/events/recent/year/
/events/recent/custom/?maxage=10&maxcount=100

Class Specific

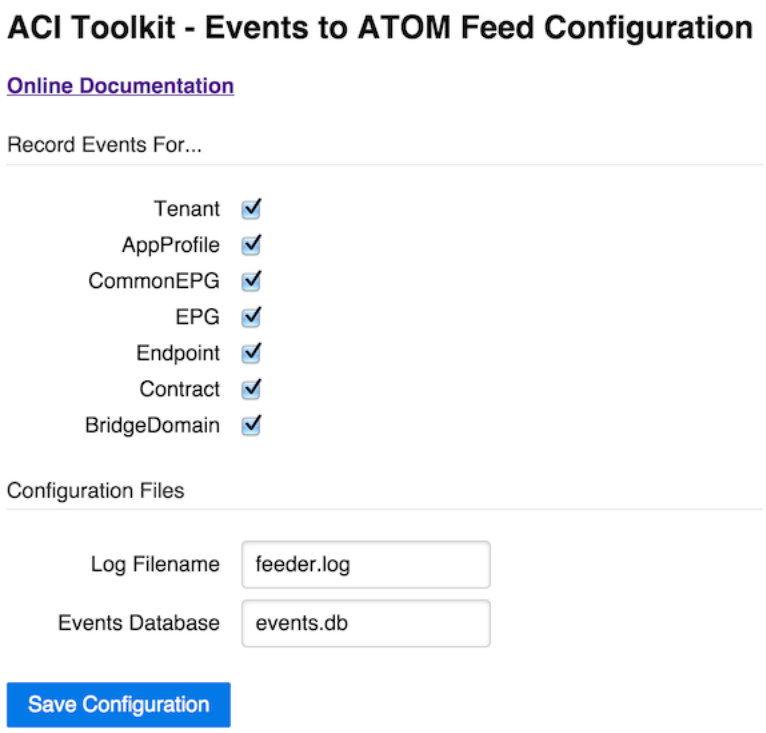
Tenant

/events/class/Tenant/
/events/class/Tenant/day/
/events/class/Tenant/week/
/events/class/Tenant/month/
/events/class/Tenant/year/
/events/class/Tenant/custom/?maxage=10&maxcount=100

AppProfile

/events/class/AppProfile/
/events/class/AppProfile/day/
/events/class/AppProfile/week/
/events/class/AppProfile/month/
/events/class/AppProfile/year/
/events/class/AppProfile/custom/?maxage=10&maxcount=100

Configuration is available at the /config/ URL e.g. *http://127.0.0.1:5000/config/*



License

Intersite Application

Overview

The Intersite application allows application policies to be applied across ACI fabrics. This is configured on an Endpoint Group (EPG) basis. When endpoints appear in an EPG that is configured as an intersite EPG, the endpoint (EP) and EPG information is propagated to the remote sites as per the intersite policy.

On the remote sites, the EP and EPG information is used at the ingress L3out interface to reclassify the endpoint traffic into the appropriate EPG. Since multiple sites may be involved, the information may be propagated to more than 1 remote site as per the intersite policy. Likewise, each site may use more than 1 L3out interface to communicate with the other sites so the endpoint and EPG information may be installed on more than 1 L3out interface within the site.

Installation

The Intersite application runs on top of the acitoolkit and comes as part of its applications suite. Installation of the acitoolkit can be performed as per the instructions here. It is recommended that the acitoolkit be installed using the `git clone` commands. Once installed, the Intersite application can be found in the directory `acitoolkit/applications/multisite`.

Usage

For each site, a single instance of the Intersite application is to be run. This instance will handle exporting the EP and EPG information from the local site to the remote sites as per the Intersite policy. Since most communication will be bidirectional, this implies that there will be a similar policy on the remote site application exporting the EPs on the desired EPGs back to the local site. The Intersite application is run as a standard python script and can be called as shown below:

```
python intersite.py -h

usage: intersite.py [-h] [--config CONFIG] [--generateconfig]
                    [--debug [{verbose,warnings,critical}]]

ACI Multisite Tool

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Configuration file
  --generateconfig      Generate an empty example configuration file
  --maxlogfiles MAXLOGFILES
                        Maximum number of log files (default is 10)
  --debug [{verbose,warnings,critical}]
                        Enable debug messages.
```

The Intersite policy that defines the operation of the tool is provided using the `--config` command line option. This option is followed by the configuration filename that contains the intersite policy. The intersite policy is described using JSON and the details are described in the *Configuration File* section.

A sample configuration file can be generated by using the `--generateconfig` command line option. Specifying this option will produce the following output and quit the application. The sample JSON will be found in the file named `sample_config.json` and the contents are described in the *Configuration File* section.

```
python intersite.py --generateconfig

Sample configuration file written to sample_config.json
Replicate the site JSON for each site.
    Valid values for use_https and local are 'True' and 'False'
    One site must have local set to 'True'
Replicate the export JSON for each exported contract.
```

The `--debug` and `--maxlogfiles` options are described in the *Logging* section.

Configuration File

The configuration file defines the policy that is used to configure the tool, connect to the local and remote APICs, and the EPG export policies. The sample configuration file generated by the `--generateconfig` command line option described in the *Usage* section is shown below.

```
{
    "config": [
        {
            "site": {
                "username": "",
                "name": "",
                "ip_address": "",
                "password": "",
                "local": "",
                "use_https": ""
            }
        },
        {
            "export": {
                "epg": "",
                "app": "",
                "tenant": "",
                "remote_epg": "",
                "remote_sites": [
                    {
                        "site": {
                            "interfaces": [
                                {
                                    "l3out": {
                                        "consumes_interface": [
                                            {
                                                "cif_name": ""
                                            }
                                        ],
                                        "name": "",
                                        "protected_by": [
                                            {
                                                "taboo_name": ""
                                            }
                                        ],
                                        "provides": [
                                            {
                                                "contract_name": ""
                                            }
                                        ],
                                        "consumes": [
                                            {
                                                "contract_name": ""
                                            }
                                        ],
                                        "tenant": ""
                                    }
                                }
                            ],
                            "name": ""
                        }
                    }
                ]
            }
        }
    ]
}
```

It should be noted that the configuration consists of site and EPG export policies.

Site Policy

The site policy defines how the tool will communicate with the APIC of each fabric. There is a single site object in the JSON configuration for each site that the tool will communicate. Each site consists of the following items:

- username: The username used to login to the APIC of this site
- name: The name of this Site used in the context of the tool to correlate which site the EPG export policy is referring.
- ip_address: The IP address of the APIC of this site
- password: The password used to login to the APIC of this site
- local: True or False. True if this site is considered the local site for this tool. Endpoints are only exported from the local site.
- use_https: True or False. True if the tool should use https to login to the APIC.

EPG Export Policy

The EPG export policy defines which EPG to export EPs and where the EPs should be exported. There is at most a single EPG export policy for any EPG. The EPG policy contains the following items:

- epg: The name of the EPG to export endpoints
- app: The name of the Application Profile that contains the EPG.
- tenant: The name of the tenant that contains the EPG.
- remote_epg: The name to be used as the EPG (l3extInstP) on the remote site outside L3 interface (l3extOut).
- remote_sites: The remote site policy for this exported EPG.

The `remote_sites` policy contains one or more remote site policies where the EPG will be exported. Each remote site policy contains the following items:

- name: The name of the remote site. This should correspond to one of the sites defined in the Site policies.
- interfaces: The list of remote L3out interface policies on the remote site to install the endpoints for this EPG.

Each L3out interface policy contains the following items which describes how the endpoints should be configured on that particular L3out interface.

- name: The name of this L3out interface. This should match the name configured in the remote site APIC.
- tenant: The tenant name that contains the L3out interface. This should match the name configured in the remote site APIC. It should be noted that this may be a different tenant than the EPG on the local site.
- provides: This contains a list of zero or more contract names that the EP should provide when configured on the remote site.
- consumes: This contains a list of zero or more contract names that the EP should consume when configured on the remote site.
- consumes_interface: This contains a list of zero or more contract interface names that the EP should consume when configured on the remote site.
- protected_by: This contains a list of zero or more taboo names that the EP should be protected by when configured on the remote site.

- noclean: True of False value that determines whether 'stale' contracts should be removed from a remote EPG on script start up (default False, stale contracts are removed).

Command Shell

When the Intersite application is running, there is a small simple command line shell that provides basic interaction with the tool. The following commands are available at the command prompt.

```
configfile <filename>     Set the configuration filename to the specified filename.
show configfile           Show the current setting of the configuration filename.
show config               Show the current JSON configuration that the tool is using.
reloadconfig              Reload the JSON configuration from the configuration file.␣
↪This is used to enable new configuration additions, changes, or deletions.
show debug                Show the current level setting of the debug messages.
debug critical            Sets the debug message level to critical.
debug warnings            Sets the debug message level to warnings.
debug verbose             Sets the debug message level to verbose.
help [cmd]                Displays help for any command.
quit                      Quit the Intersite tool.
```

REST API

The Intersite application can be accessed through a simple server application using a REST API. When run as a server, the application usage is run as follows:

```
python intersite_rest_server.py -h
usage: intersite_rest_server.py [-h] [--config CONFIG]
                                [--maxlogfiles MAXLOGFILES] [--generateconfig]
                                [--debug [{verbose,warnings,critical}]]

ACI Multisite Tool

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Configuration file
  --maxlogfiles MAXLOGFILES
                        Maximum number of log files (default is 10)
  --generateconfig      Generate an empty example configuration file
  --debug [{verbose,warnings,critical}]
                        Enable debug messages.
  --ip IP               IP address to listen on.
  --port PORT           Port number to listen on.
```

An example of running the application would be:

```
python intersite_rest_server.py --config my_config.json
```

By default, the server runs on the loopback IP address 127.0.0.1 and the port 5000. This can be changed via the `--ip` and `--port` command line options.

The REST API allows retrieving and setting the configuration of the tool. This is done through a single URL, namely `/config`.

By sending HTTP GET requests to `/config`, the configuration can be retrieved.

By sending JSON configuration in an HTTP POST or HTTP PUT to `/config`, the configuration can be changed. This JSON configuration is the same format as decribed in the *Configuration File* section. If the configuration is not valid JSON, a 400 error response code will be returned.

The REST API is protected by basic HTTP authentication with the default username set to `admin` and the default password set to `acitoolkit`.

Some examples using the REST API via curl:

```
curl -i -u admin:acitoolkit -H "Content-Type: application/json" -X PUT -d@my_config.
↪json http://localhost:5000/config
curl -i -u admin:acitoolkit -X GET http://localhost:5000/config
```

Automator

The intersite application can be run in a basic automated fashion which will identify EPG's with an assigned tag and automatically create the appropriate L3out objects and contracts from the templated configuration. The usefulness of this utility is limited by the requirement of tenant's needing to exist in both sites (APIC's) consistantly.

```
python intersite_automator.py -h
usage: intersite_automator.py [-h] [--config CONFIG] [--generateconfig]
                              [--debug [{verbose,info,warnings,critical}]]
                              [--stdout]

ACI Multisite Automation Tool

optional arguments:
   -h, --help            show this help message and exit
  --config CONFIG       Configuration file in JSON format
  --generateconfig      Generate an empty example configuration file
  --debug [{verbose,info,warnings,critical}]
                        Enable printing of debug messages
  --stdout              Output all log events to stdout
```

To make use of this modified version of the intersite application, you will need to use a different configuration file. The automator removes the need for statically defining the individual export policies but requires a new 'automator' specific configuration.

```
{
"config": [
    {
        "site": {
            "name": "",
            "username": "",
            "password": "",
            "ip_address": "",
            "use_https": "",
            "local": ""
        }
    },
    {
        "site": {
            "name": "",
            "username": "",
            "password": "",
            "ip_address": "",
            "use_https": "",
```

```
            "local": ""
        }
    }
],

"automator": {
    "check_interval": "60",
    "search_filter": "replicated",

    "remote_l3out": {
        "tenant": "Layer3OutTenant",
        "interface_name": "L3Out.Site1-Site2.SERVER",
        "network_name": "Site2.%{tenant}.%{app}.%{epg}"
    },

    "remote_contracts": {
        "consume_contract": [{
            "name": "CT.%{tenant}.%{app}.%{epg}.to.Site2",
            "default_filter": "east-west-allow-all",

            "export_to_epg_owner": "True",
            "export_name": "x.CT.%{tenant}.%{app}.%{epg}.to.Site2"
        }],
        "provide_contract": [{
            "name": "CT.%{tenant}.%{app}.%{epg}.to.Site2",
            "default_filter": "east-west-allow-all",

            "export_to_epg_owner": "True",
            "export_name": "x.CT.%{tenant}.%{app}.%{epg}.to.Site2"
        }],
        "consume_int_contract": []
    }
}
}
```

The site specific configuration hasn't changed, and matches the definition in the previous section of this documentation

- check_interval: integer - The time in seconds that intersite_automator will sleep between searching for EPG's with the search_filter tag (this should be a >0 value to reduce API calls on the APIC)
- search_filter: string - The tag that intersite_automator will search for (on EPG's) to identify which EPG's to replicate.
- remote_l3out->tenant: string - The tenant that owns the L3out object that will have the 'remote EPGs' created under.
- remote_l3out->interface_name: string - The interface that will have the 'remote EPGs' created under.
- remote_l3out->network_name: string - The network name pattern (see pattern information below) that will be used to create the new remote EPG.
- remote_contracts-><contract type>->name: string - The contract name pattern (see pattern information below) that will be attached to the newly created remote EPG (this will be created if it does not exist)
- remote_contracts-><contract type>->default_filter: string - A default filter that will be attached to a contract that is created by the intersite automator
- remote_contracts-><contract type>->export_to_epg_owner: boolean string - Determines whether or not the intersite automator script should export this contract to the EPG owner. This is useful if your L3out object sits in a different tenant as the EPG but also assumes that there is some level of consistancy across multiple APICs.

- remote_contracts-><contract type>->export_name: string - The pattern to use when exporting the contract to the EPG owner.

Patterns Certain configuration fields allow the use of a very primitive variable substitution. This enables you to create EPGs and Contracts with dynamic names. The following variables are currently supported:

- ${tenant} - The name of the tenant that owns the EPG
- ${app} - The name of the application that the EPG belongs to
- ${epg} - The name of the EPG that is being replicated

Logging

The Intersite application supports logging of various debug messages. The messages are categorized by levels and the application allows the level to be set by the user through the `--debug` option. The lowest level is `critical` and will only log the most critical messages. The next level is `warnings` and will log all of the `critical` messages as well as those deemed suspect. The `verbose` level gives the previous levels plus additional information that provides deep insight into what has occurred in the tool.

The debug messages are stored in the file `intersite.log`. When the file reaches 5 MB in size, the log file will rollover to a new file up to the configured maximum number of log files. By default, this maximum is 10 but the number can be changed by the user via the `--maxlogfiles` command line option. Once the maximum number of log files has been reached, the oldest log file will be deleted.

APIC Object Model

The following objects are used from the APIC Object Model to enable the functionality of the Intersite tool.

The EPG that is configured to have its endpoints exported to the remote sites is defined using the classes fvTenant, fvAp, and fvAEPg.

The endpoints are tracked by subscribing to notifications from the fvCEp and fvStCEp classes and examining the EPG membership of those endpoints.

The endpoints are installed on the remote site by using the classes l3extInstP and l3extSubnet under l3extOut. The endpoint address is installed as a /32 entry within the l3extSubnet. The l3extInstP contains the children fvRsProv, fvRsCons, fvRsConsIf, and fvRsProtBy to associate the endpoint with the appropriate contracts, taboos, and contract interfaces.

Additionally, a tagInst object is placed on every l3extInstP that contains specially formatted string within the opaque tagInst name field. This string is formatted as follows: **isite**:*tenant_name*:*app_name*:*epg_name*:*site_name* where the tenant, app, and epg names are that where the original endpoint is connected and the site_name is that of the site that installed the l3extInstP.

Importing within other applications

Each instance of the tool is represented by the Collector class. Within the Collector class, there will be multiple instances of the Site class, namely the LocalSite and RemoteSite classes.

If you wish to include the Intersite application in your own app, the policies can be defined as python dictionaries directly and passed to the LocalSite instance using the `add_policy()` function. This will allow the Monitor class to begin exporting the EP of the specified EPG according to the policy. Similarly, the external configuration file can be updated and the `reload_config()` function can be called within the Collector class.

Connection Search : Configured Connection Search Tool

Summary

Connection Search is an application that allows the user to search for connections between endpoints on the ACI fabric.

General Overview

The Connection Search GUI provides a simple search bar. The user can enter a query into this search bar and the results are then displayed.

The results are in the format of a table where set of IP addresses or subnets in the first column corresponds to connection sources, a set of IP addresses or subnets in the second column corresponds to connection destinations, and a final column that shows which filters are applied to these connections.

Each row of the table is for a unique source-dest combination.

A search query consists of a set of *attribute=value* tuples. Any attribute can be used at most one time in a given search. If an attribute is not specified, then the search will assume those omitted attributes should not be used to qualify the results. For example, if NO attributes are specified, then ALL connections will be reported. If *sip=10.11.12/24* is specified, then only connections that have a source that is covered by the subnet *10.11.12.0/24* will be reported.

Installation

Connection Search is part of the acitoolkit and will be installed at the same time. The installation details can be found here.

Web based Usage

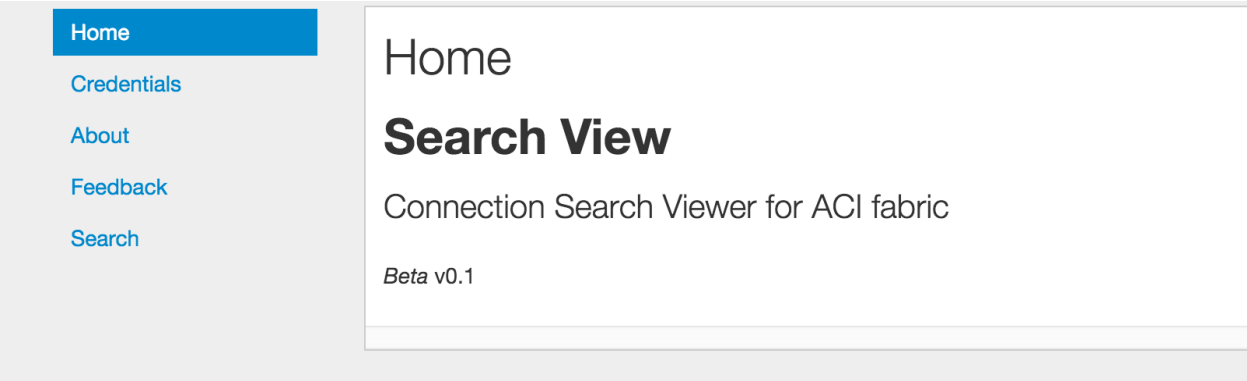
Connection Search should be run as a web application. Running Connection Search as a web application is done by switching to the connection_search directory and running the application:

```
cd acitoolkit/applications/connection_search
python aciConSearchGui.py
```

By default, Connection Search will run locally on the loopback address. Accessing the tool is done by running a web browser locally and pointing to the following URL:

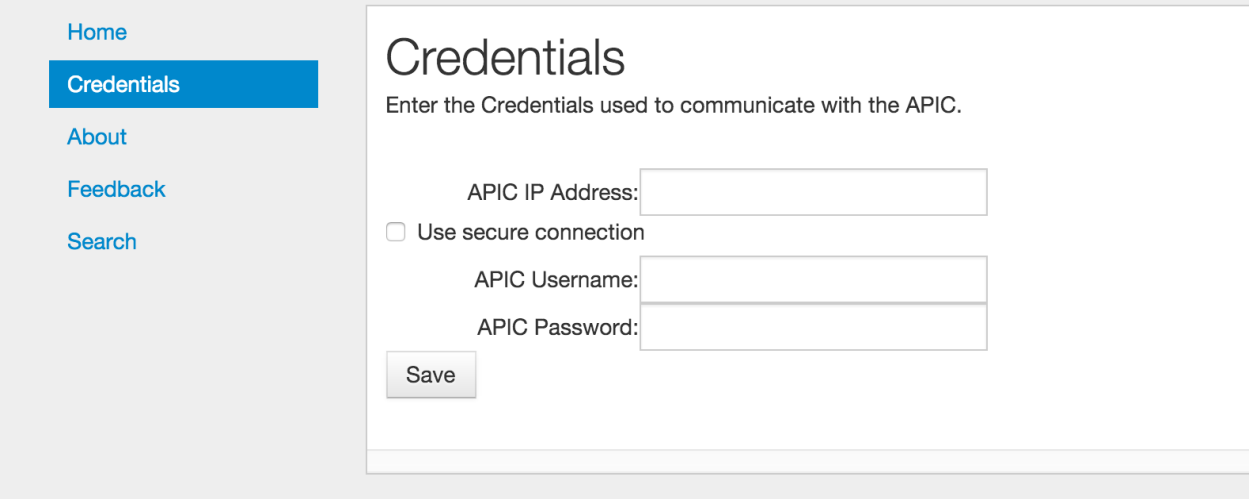
```
http://127.0.0.1:5001/
```

Upon pointing the web browser to the Connection Search URL, the Connection Search welcome screen along with the navigation menu on the left hand side will be displayed.



Credentials

Entering the APIC credentials is necessary for Connection Search to be able to perform searches. Selecting the Credentials in the navigation menu will bring up the Credentials screen where the APIC credentials can be entered.



A number of fields allow the entry of the APIC credentials. These fields are:

APIC IP Address: This field is the IP address used to communicate with the APIC.

Use secure connection: When selected, this checkbox indicates that the APIC communication uses https instead of http.

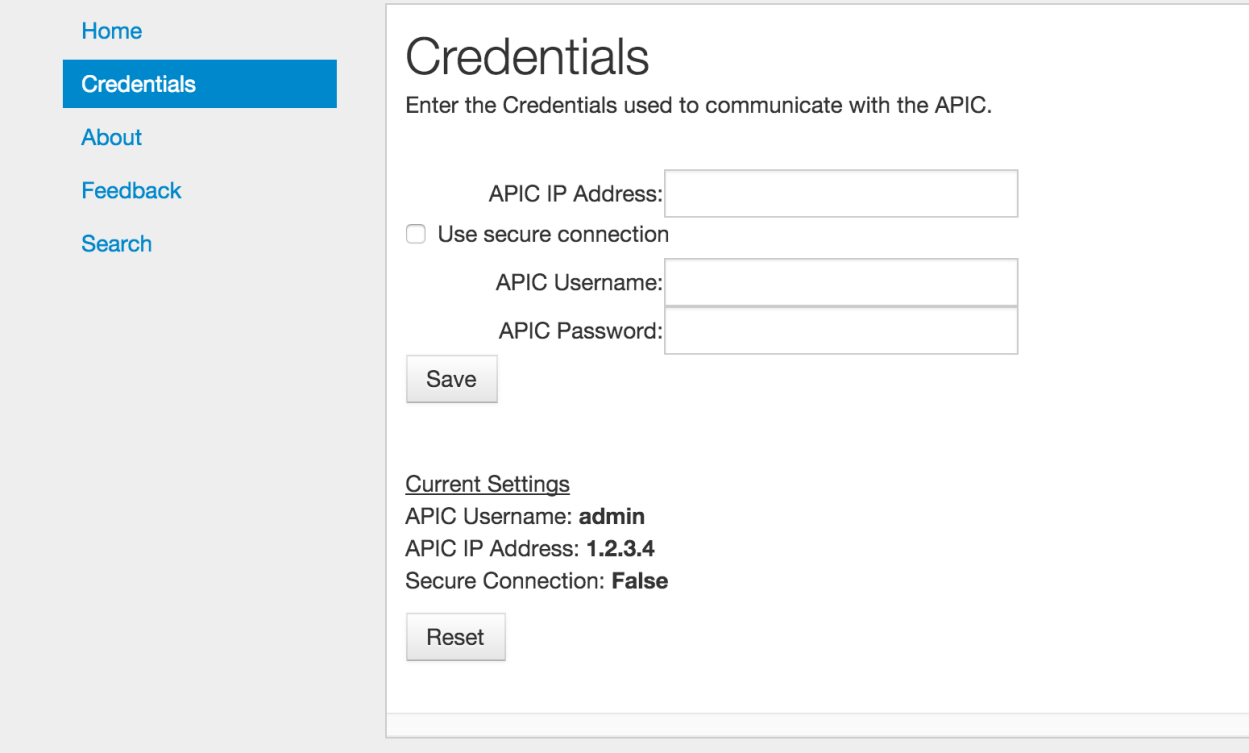
APIC Username: The username used when logging in to the APIC.

APIC Password: The password used when logging in to the APIC.

The credentials are stored with Connection Search by selecting the **Save** button. Once the credentials are entered, they will be saved as part of the user session on that browser. This means that another window or tab using Connection Search can be opened on the same machine without having to re-enter the credentials.

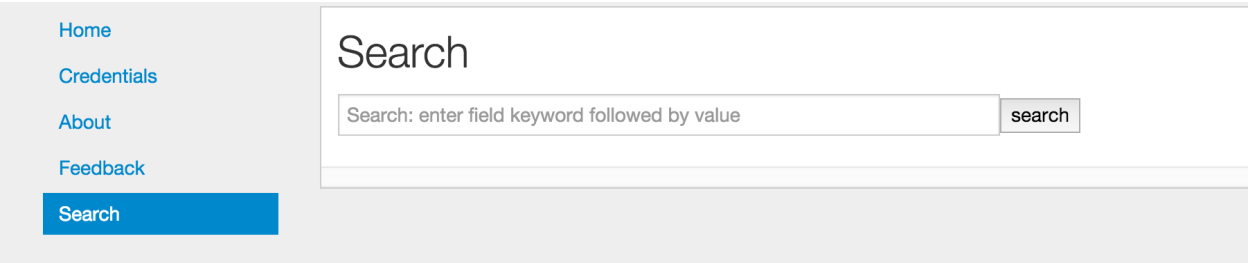
When entered, the current credentials are shown as in the screenshot below. Note that for security purposes, the password is not displayed. Clicking the **Reset** button will cause Connection Search to forget the current credentials,

but will not impact the APIC in any way.



Performing a Search

Selecting **Search** from the navigation menu will bring up the main search screen. Note that when this item is initially selected the Connection Search application will read in the configuration of the APIC. This may take some time depending upon the size of your configuration as well as the speed of your network connection. Your browser may indicate that it is "waiting on 127.0.0.1" while the configuration is loading. Please be patient.



A search consists of entering the search filter criteria in the search bar and then pressing the **SEARCH** button. If no criteria is entered, all connections will be displayed.

The search criteria consists of a set of *attribute=value* tuples separated by spaces. Each attribute can only be used once. If it is used more than once, the last one is the only one that will take effect.

The Attributes are as follows:

> **tenant**: The name of the tenant to conduct the search in. Note that this will be the tenant used for both the source and the destination so it will not find connections that are end-points using services shared from a different tenant. A search that does not specify the tenant should be used when looking for these kinds of connections.

The tenant name can also contain a simple "*" as a wild-card. For example, *tenant=foo** will match on tenants with the name *foo*, *foobar*, and *foo_xyz*. *tenant=foo*bar* will match on tenants with the name *foobar* and *foohellobar*, but not *foobar5*.

context: The name of the context to conduct the search in. Like the *tenant* attribute, this one will apply to both the source and destination and can contain wild-cards. The *context* attribute would typically be used in conjunction with the *tenant* attribute, but that is not required.

contract: The name of a contract can be specified and only connections that use contract(s) with that name will be selected. Note that the contact name cannot be further qualified with the tenant name. However, the tenant name can be one of the attributes, but that will limit the result to only endpoints that are under that tenant.

sip: The source IP address or source subnet. When entering an address, it should be in the defacto standard form of *a.b.c.d*. For example, 10.2.5.8. When entering a subnet, it should take the form of *a.b.c.d/s* where "s" is the length of the subnet mask. Any of the prefix digits that are omitted are assumed to be zero. For example, *192/8* is equivalent to *192.0.0.0/8* and *192/16* is equivalent to *192.0.0.0/16*. The *sip* attribute does not accept the "*" wild-card.

Note that when doing a search where the matching criteria can be either more or less specific than the field in the policy, the result displayed with be value in the policy. For example, if a search is done for *sip=1.2.3.0/24* and there is a L3Out that has the address *1.2/16* that matches that *sip*, the output result will show *1.2/16*. Similarly, if there is an EPG with addresses *1.2.3.4/32* and *1.2.3.89/32*, the result displayed in the source column will be *1.2.3.4/32* and *1.2.3.89/32*.

dip: The destination IP address or destination subnet. See **sip** above for how this attribute works.

dport: The layer 4 destination port or port range. This attribute implies that the *prot* field is *tcp udp* and the *etherT* field is *ip* unless otherwise specified. The format is either a single value or a range. When entering a range, the minimum value should be separated from the maximum value by a dash "-". For example, *dport=80* or *dport=20-45*. A limited set of common protocol acronyms can also be used. The currently supported set is:

'http', 'https', 'ftp-data', 'smtp', 'dns', 'pop3', 'rtsp', and 'any'

sport: The layer 4 source port or port range. This attribute works just like the *dport* attribute above.

prot: The layer 4 protocol type. This attributes allows the user to select which protocol to search for. Possible values are:'icmp', 'igmp', 'tcp', 'egp', 'igp', 'udp', 'icmpv6', 'eigrp', 'ospfigp', 'pim', 'l2tp' and 'any'.

If the *prot* attribute is specified, then the *etherT* attribute is assumed to be *ip* unless otherwise specified. If the *prot* attribute is specified as *any*, then the *etherT* is not assumed. Note, that there is really no reason to specify *prot=any* as this is the default anyway.

etherT: The Ethertype of the protocol. Typical values for the *etherT* attribute are *ip* or *arp*. Specific numeric values can also be used, however they will be matched against the actual string value provided by the APIC filterEntry object, so a value of *800* will not match *ip*.

The *etherT* field is assumed to be *ip* if the *prot* field is specified to be an IP protocol. It is assumed to be *arp* if the *arpOpc* attribute is specified.

arpOpc: The ARP opcode. Possible values are *req*, *reply* and *any*. The default is *any*. When the *arpOpc* is specifed, the *etherT* is assumed to be *arp*.

applyToFrag: Apply to fragments attribute. Possible values are *yes*, *no* and *any*. The default is *any*.

tcpRules: TCP rules. This allows the user to specify the TCP rules to match. Use of this field in Connection Search has not been fully validated and so should be used with caution (it will do no harm so you may play around with it, just be aware the results may not be what you expect). It must be entered exacty

the way the APIC specifies it the FilterEntry object. If *tcpRules* are specified, then the *prot* attribute is assumed to be *tcp*.

When the search results are displayed, placing the cursor over a table cell will cause a fully qualified name of the EPG or OutsideEPG to be displayed and the contract.

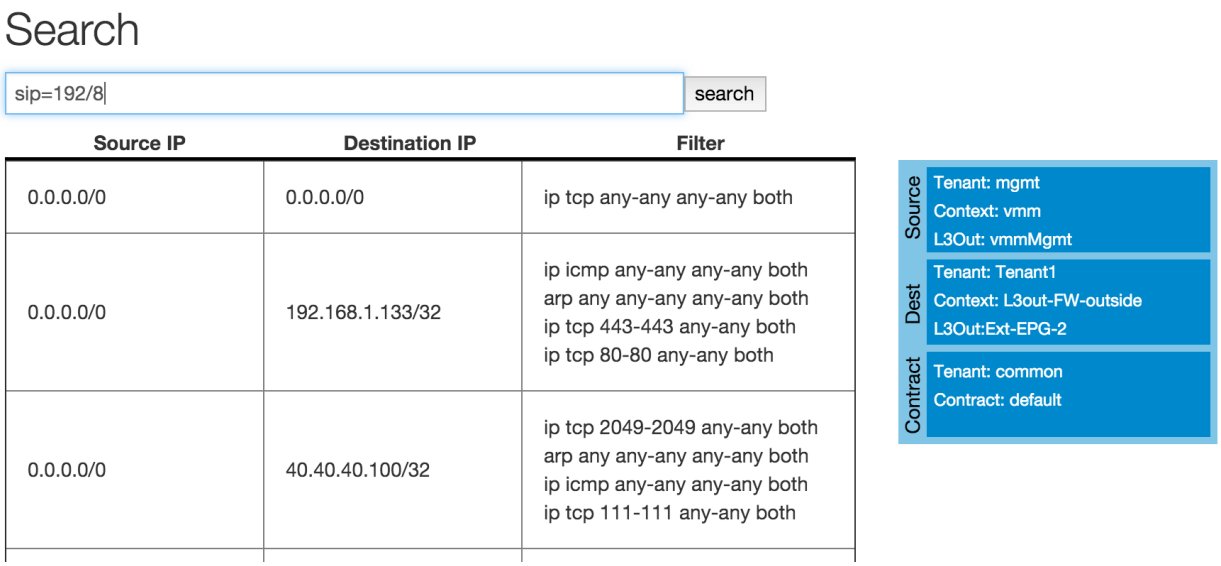
For communication within an EPG, i.e. between two end-points that are in the same EPG, Connection Search will create an "implied" contract that is both provided and consumed by that EPG. The filter in that contract will allow all communication. When the cursor hovers over the filter cell for such a connection, the contract name will begin with "implied" followed by a number that makes it unique. This contract does not actually exist in the APIC.

Examples

The following are a few examples of searches and explanation of the results.

Example 1

Find all the connections whose source IP address is in the subnet "192.0.0.0/8".



Here we see how the search is contructed, `sip=192/8`. This tells the application to find all connections whose source IP address has the first 8-bits equal to "192".

The results show three connections. The first shows traffic from `0.0.0.0/0` to `0.0.0.0/0`. This shows that all traffic from the `mgmt` tenant can be switched to `Tenant1 L3Out-FW-outside` as long as it is `TCP` traffic. It has a filter of `ip tcp any-any any-any both` which should be interpreted as:

- EtherType = IP
- IP protocol = TCP
- L4 destination port minimum = any
- L4 destination port maximum = any
- L4 source port minimum = any
- L4 source port maximum = any
- Direction with respect to destination = both

The following two rows show connections from `Tenant1` to `Tenant1`, i.e. they correspond to different traffic than the first row.

Note that there were additional results that are not shown in the above image.

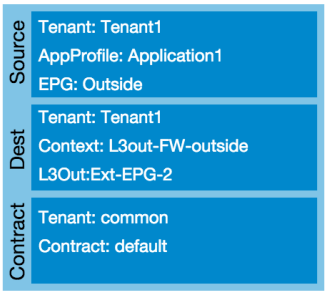
Example 2

This next example shows a search looking for traffic in a tenant named "Tenant1", whose destination is to anything in the `192.168.0.0/16` subnet on any layer 4 destination port number in the range of `80` to `500`.

Search

tenant=Tenant1 dip=192.168/16 dport=80-500 search

Source IP	Destination IP	Filter
0.0.0.0/32	0.0.0.0/0	ip tcp any-any any-any both
0.0.0.0/0	192.168.1.133/32	ip tcp 443-443 any-any both ip tcp 80-80 any-any both
192.168.1.133/32	0.0.0.0/0	ip tcp any-any any-any both
192.168.1.133/32	192.168.1.133/32	any any any-any any-any both any any any-any any-any both



The results show 3 connection groups. The first one is from a host IP of `0.0.0.0/32`. This looks funny and is probably a configuration error in the APIC. The destination here is `0.0.0.0/0` and the filter is for any TCP traffic. `0.0.0.0/0` in the destination column covers the search criteria of `dip=192.168.0.0/16`, and the filter of any-any for the destination port range covers the search criteria of `dport=80-500`.

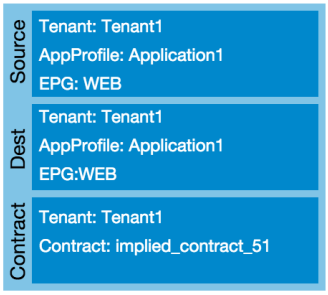
Example 3

In this query, the user wants to see all of the destinations that host `192.168.1.133` in tenant "Tenant1" can send traffic to.

Search

sip=192.168.1.133/32 tenant=Tenant1 search

Source IP	Destination IP	Filter
192.168.1.133/32	0.0.0.0/0	ip tcp any-any any-any both
192.168.1.133/32	192.168.1.133/32	any any any-any any-any both



Here there are two results. The first shows that this host can send traffic to default route of `0/0` for any TCP traffic. The second row shows that this host can communicate with itself using any protocol. This second one has a fully open filter of `any any any-any any-any both`. When we place the cursor over this row we can see that this was an "implied" filter. The implied filter is created by the connection search tool to show that end-points within an EPG can communicate with each other without any constraint.

About

This is a simple summary screen with a link to the documentation and a description of the software license.

Feedback

This screen provides the ability for the users to submit comments, suggestions, feature requests, and bug reports directly to the authors of the tool.

Command Line Usage

Connection Search can also be run as a command line application. This is done by switching to the Connection Search directory and running the application:

```
cd acitoolkit/applications/connection_search
```

The application is run with options provided in the command line. The help for this command is shown by:

```
python aciConSearch.py --help
```

The output for this command is shown below:

```
usage: aciConSearch.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD]
                       [--snapshotfiles SNAPSHOTFILES [SNAPSHOTFILES ...]]
                       [-tenant TENANT] [-context CONTEXT] [-sip SIP]
                       [-dip DIP] [-dport DPORT] [-sport SPORT]
                       [-etherT ETHERT] [-prot PROT] [-arpOpc ARPOPC]
                       [-applyToFrag APPLYTOFRAG] [-tcpRules TCPRULES]

Connection Search tool for APIC.

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     APIC IP address.
  -l LOGIN, --login LOGIN
                        APIC login ID.
  -p PASSWORD, --password PASSWORD
                        APIC login password.
  --snapshotfiles SNAPSHOTFILES [SNAPSHOTFILES ...]
                        APIC configuration files
  -tenant TENANT        Tenant name (wildcards, "*", accepted), default "*"
  -context CONTEXT      Context name (wildcards, "*", accepted), default "*"
  -sip SIP              Source IP or subnet - e.g. 1.2.3.4/24, default: "0/0"
  -dip DIP              Destination IP or subnet - e.g. 1.2.3.4/24, default:
                        "0/0"
  -dport DPORT          Destination L4 Port value or range, e.g. 20-25 or 80.
                        Default: "any"
  -sport SPORT          Source L4 Port value or range, e.g. 20-25 or 80.
```

```
                        Default: "any"
  -etherT ETHERT        EtherType, e.g. "ip", "arp", "icmp". Default: "any"
  -prot PROT            Protocol, e.g. "tcp", "udp". Default: "any"
  -arpOpc ARPOPC        ARP Opcode, e.g. "req", "ack". Default: "any"
  -applyToFrag APPLYTOFRAG
                        Apply to fragment, e.g. "yes", "no". Default: "any"
  -tcpRules TCPRULES    TCP rules, e.g. "syn", "fin". Default: "any"
```

The APIC credentials are provided with the *–url*, *–login*, and *–password* options.

The remaining fields follow the attributes described above for the GUI version of the tool. Note that when aciConSearch.py is used from the command line in this manner, the APIC configuration will be loaded and a single search conducted. When the GUI version is used, the APIC configuration is loaded and multiple searches can be made against it without doing a re-load.

Fake APIC

Purpose

- The Fake APIC is designed for users to view Managed Objects (and their properties) based on JSON configuration files
- The Fake APIC works as an **offline-tool** for users who may not have access to the APIC, but still want to see certain (or all) Managed Objects on the network.

Usage

1. To generate JSON configuration files, use the snapback application located at `acitoolkit/applications/snapback`
2. Run the `aciconfigdb.py` file

```
python aciconfigdb.py -u <APIC url> -l <login> -p <password> -s --v1 -a
```

- The `-s` option takes the snapshot of the configuration from the APIC
- The `--v1` option takes the snapshot using direct HTTP queries rather than the configuration import and export policies. This is important to be able to simulate HTTP responses.
- The `-a` option ensures the configuration includes all properties of the class objects
  - It's **very important** to give the `-a` because the Fake APIC depends on the all properities of the class objects
- The JSON files will be located at: `acitoolkit/applications/snapback/apic-config-db`
- Depending on the APIC, getting all the data may take around 5 - 8 seconds
- Modify a sample file such as `aci-show-epgs.py` to use the FakeSession class from the Fake APIC
- Pass in the JSON files to the FakeSession constructor

```
from os import listdir
from acitoolkit.acifakeapic import FakeSession
...
...
# Set the directory to the location of the JSON files
```

```
directory = 'applications/snapback/apic-config-db/'
filenames = [directory + file for file in listdir(directory)
             if file.endswith('.json')]

# Create the session
session = FakeSession(filenames)
```

- Run the file

```
python aci-show-epgs.py -u <dummy url> -l <dummy login> -p <dummy password>
```

- Since this file will be using the Fake APIC, you can pass in *any* value for the url, login, and password
- The Fake APIC works by retrieving data from the JSON files to mimick responses from the real APIC
- Users can pass in queries such as `/api/mo/uni/tn-tenant1/BD-Br1.json?query-target=children` to get back responses.

How to pass in queries to the Fake APIC

```
...
...
# to print the data nicely
import json

session = FakeSession(filenames)
query = '/api/mo/uni/tn-tenant1/BD-1.json?query-target=children'
fake_ret = fake_session.get(query)
fake_data = fake_ret.json()['imdata']
data = fake_ret.json()['imdata']
# print the data from the Fake APIC
print json.dumps(data, indent=4)
```

What queries the Fake APIC supports

- Any queries starting with `api/mo/` can have the **scoping filters** of
- query-target
- rsp-subtree
- target-subtree-class
- Queries starting with `api/node/class` can only have the `query-target` with the value of `self`
  - values of rsp-subtree and target-subtree-class are supported
- For more information regarding **scoping filters** see page 12 of the Cisco APIC REST API User Guide under the section "Applying Query Scoping Filters"

Dependencies

- Python 2.7
- Data in the JSON configuration files
- The Fake APIC can **only** retrieve data that are in the JSON files, it cannot retrieve any data from the real APIC

- The Fake APIC does **not** check for bad queries

ACI Reports

The `ACI Reports` is a collection of reporting tools for the APIC logical and physical model.

ACI ReportView GUI

Reports can be generated using a GUI based application. Switch reports or Tenant reports can be generated using this application.

Installation

This reporting application is included in the `acitoolkit` package when downloaded using the `git clone` method of installation. The application can be found in the `acitoolkit/applications/reports` directory.

Usage

The application is started from the command line. In its simplest form, it can be invoked by the following command:

```
python aciReportGui.py
```

The full command help is shown below:

```
python aciReportGui.py -h
usage: aciReportGui.py [-h] [--ip IP] [--port PORT] [--test]
                       [--debug [{verbose,warnings}]]

ACI Report Viewer Tool.

optional arguments:
  -h, --help            show this help message and exit
  --ip IP               IP address to listen on.
  --port PORT           Port number to listen on.
  --test                Enable functions for lab testing.
  --debug [{verbose,warnings}]
                        Enable debug messages.
```

By default, the reporting application runs on port `5000`. This port can be changed by passing the `--port` command line option.

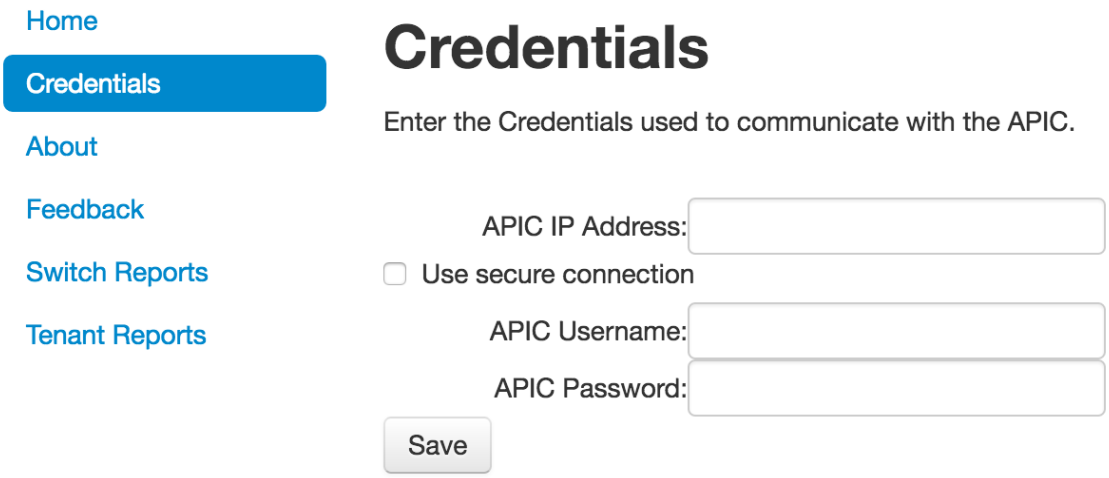
By default, the IP address used is the `127.0.0.1` loopback address. This allows the application to be accessed locally. If the application needs to be accessed from web browsers on different machines, it should be run on the actual IP address of the local machine. This can be specified using the `--ip` command line option.

Once invoked, the GUI is displayed by connecting to `http://127.0.0.1:5000` (assuming that the IP and Port are using the default values).

Credentials

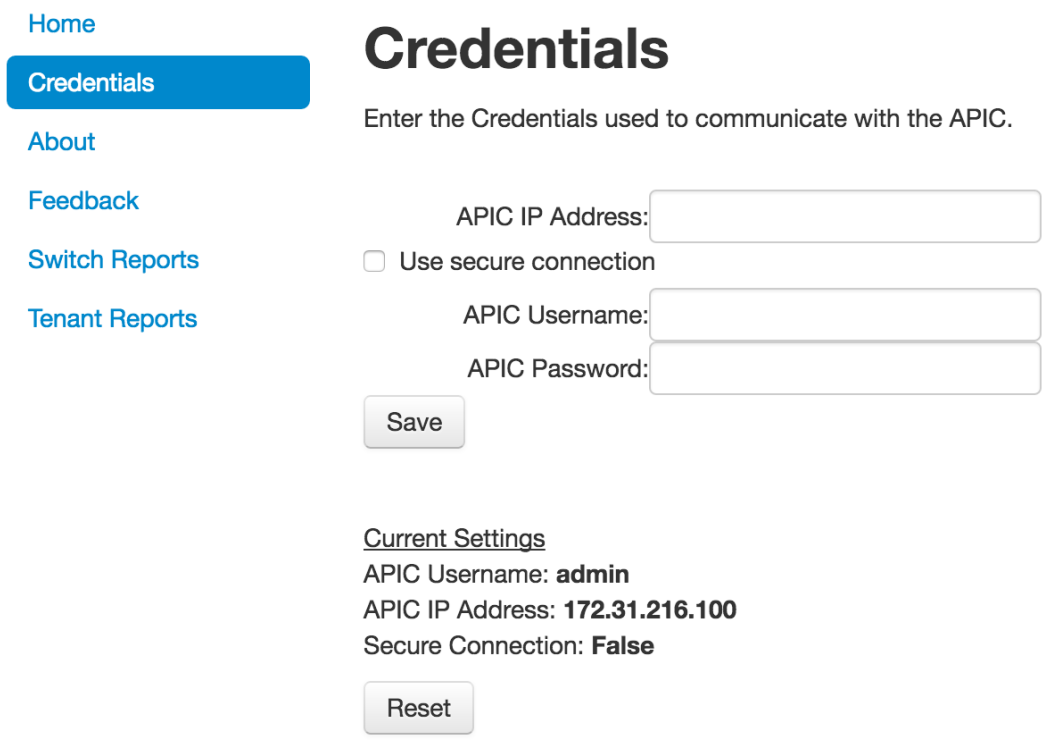
Before the application can generate reports, the application needs to have the credentials required to login to the APIC. This is done by selecting the `Credentials` menu item in the main ReportsView application.

Selecting the `Credentials` menu item will display the following form for entering credentials.



The IP address, username, and password of the desired APIC are entered. If the APIC requires HTTPS, the `Use secure connection` checkbox should be selected.

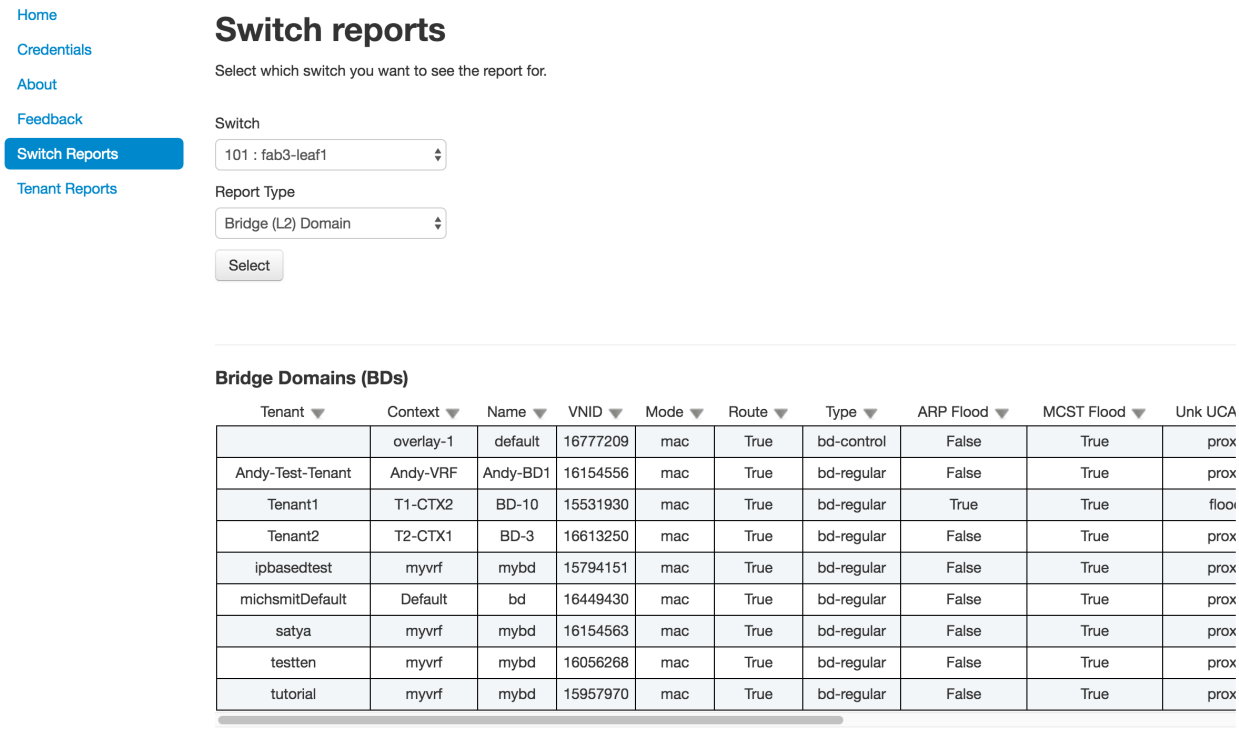
Once the credentials have been entered and the `Save` button is cliecked, the credentials will be displayed as having been set.



The credentials can be purged by clicking on the `Reset` button.

Switch Reports

Once credentials have been entered, the switch reports can be displayed by selecting the `Switch Reports` menu option. Depending on the size of the ACI fabric, it may take a few seconds for the switch reports to load.



Tenant	Context	Name	VNID	Mode	Route	Type	ARP Flood	MCST Flood	Unk UCA
	overlay-1	default	16777209	mac	True	bd-control	False	True	prox
Andy-Test-Tenant	Andy-VRF	Andy-BD1	16154556	mac	True	bd-regular	False	True	prox
Tenant1	T1-CTX2	BD-10	15531930	mac	True	bd-regular	True	True	floo
Tenant2	T2-CTX1	BD-3	16613250	mac	True	bd-regular	False	True	prox
ipbasedtest	myvrf	mybd	15794151	mac	True	bd-regular	False	True	prox
michsmitDefault	Default	bd	16449430	mac	True	bd-regular	False	True	prox
satya	myvrf	mybd	16154563	mac	True	bd-regular	False	True	prox
testten	myvrf	mybd	16056268	mac	True	bd-regular	False	True	prox
tutorial	myvrf	mybd	15957970	mac	True	bd-regular	False	True	prox

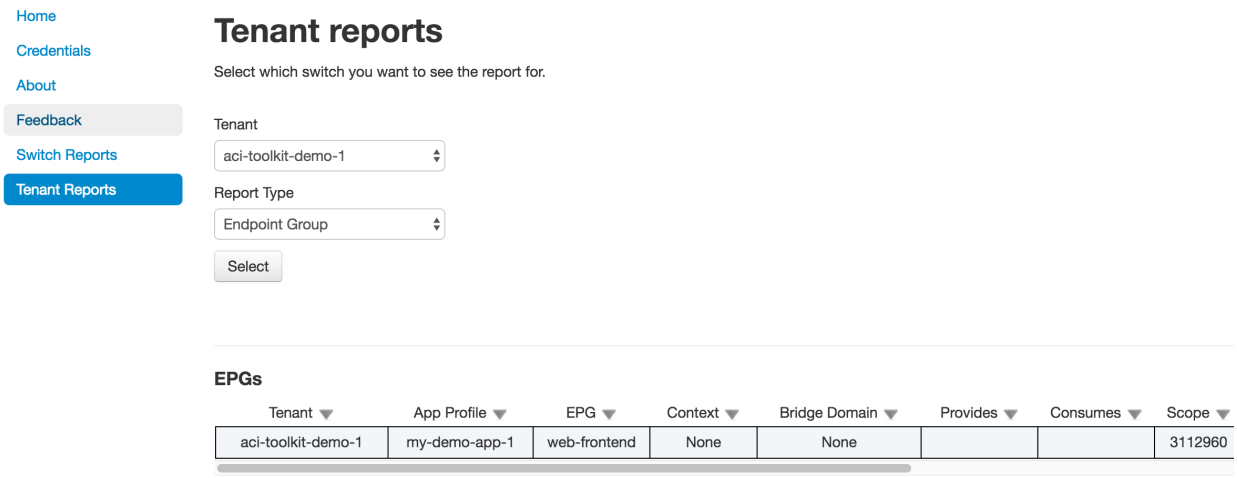
The switch report screen contains a pulldown menu to select the individual switch to provide a report.

There are a number of report types that are available. These are selected by using the `Report Type` pulldown menu. The available report types are:

- Basic
- Supervisor Card
- Linecard
- Power Supply
- Fan Tray
- Overlay
- Context
- Bridge (L2) Domain
- Endpoint
- SVI (Router Interface)
- Access Rule
- ARP
- Port Channel (incl. VPC)

Tenant Reports

Once credentials have been entered, the tenant reports can be displayed by selecting the `Tenant Reports` menu option. Depending on the size of the ACI fabric, it may take a few seconds for the tenant reports to load.



The tenant report screen contains a pulldown menu to select the individual tenant to provide a report.

There are a number of report types that are available. These are selected by using the `Report Type` pulldown menu. The available report types are:

- Context
- Bridge Domain
- Contract
- Taboo
- Filter
- Application Profile
- Endpoint Group
- Endpoint

ACI Logical Reports

Logical Tenant level reporting can be dome via this command line application.

Installation

This reporting application is included in the `acitoolkit` package when downloaded using the `git clone` method of installation. The application can be found in the `acitoolkit/applications/reports` directory.

Usage

The application is started from the command line. In its simplest form, it can be invoked by the following command:

```
python aci-report-logical.py
```

The full command help is shown below:

```
python aci-report-logical.py -h

usage: aci-report-logical.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD]
                             [-t TENANT] [-all] [-basic] [-context]
                             [-bridgedomain] [-contract] [-taboo] [-filter]
                             [-app_profile] [-epg] [-endpoint]

Simple application that logs on to the APIC and displays reports for the
logical model.

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     APIC URL e.g. http://1.2.3.4
  -l LOGIN, --login LOGIN
                        APIC login ID.
  -p PASSWORD, --password PASSWORD
                        APIC login password.
  -t TENANT, --tenant TENANT
                        Specify a particular tenant name
  -all                  Show all detailed information
  -basic                Show basic tenant info
  -context              Show Context info
  -bridgedomain         Show Bridge Domain info
  -contract             Show Contract info
  -taboo                Show Taboo (Deny) info
  -filter               Show Filter info
  -app_profile          Show Application Profile info
  -epg                  Show Endpoint Group info
  -endpoint             Show End Point info
```

Notes

The reporting application can generate a large amount of data. It may take some time to collect all of the data depending on the size of the ACI fabric. This is especially true when executing the `-all` command line option.

ACI Switch Reports

Switch level reporting can be dome via this command line application.

Installation

This reporting application is included in the `acitoolkit` package when downloaded using the `git clone` method of installation. The application can be found in the `acitoolkit/applications/reports` directory.

Usage

The application is started from the command line. In its simplest form, it can be invoked by the following command:

```
python aci-report-switch.py
```

The full command help is shown below:

```
python aci-report-switch.py -h

Simple application that logs on to the APIC and displays reports for the
switches.
usage: aci-report-switch.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD] [-s SWITCH]
                            [-all] [-basic] [-linecard] [-supervisor]
                            [-fantray] [-powersupply] [-arp] [-context]
                            [-bridgedomain] [-svi] [-accessrule] [-endpoint]
                            [-portchannel] [-overlay] [-tablefmt TABLEFMT]

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     APIC URL e.g. http://1.2.3.4
  -l LOGIN, --login LOGIN
                        APIC login ID.
  -p PASSWORD, --password PASSWORD
                        APIC login password.
  -s SWITCH, --switch SWITCH
                        Specify a particular switch id, e.g. "102"
  -all                  Show all detailed information
  -basic                Show basic switch info
  -linecard             Show Lincard info
  -supervisor           Show Supervisor Card info
  -fantray              Show Fantray info
  -powersupply          Show Power Supply info
  -arp                  Show ARP info
  -context              Show Context (VRF) info
  -bridgedomain         Show Bridge Domain info
  -svi                  Show SVI info
  -accessrule           Show Access Rule and Filter info
  -endpoint             Show End Point info
  -portchannel          Show Port Channel and Virtual Port Channel info
  -overlay              Show Overlay info
  -tablefmt TABLEFMT    Table format [fancy_grid, plain, simple, grid, pipe,
                        orgtbl, rst, mediawiki, latex, latex_booktabs]
```

Notes

The reporting application can generate a large amount of data. It may take some time to collect all of the data depending on the size of the ACI fabric. This is especially true when executing the `-all` command line option.

ACI Security Report

This application provides simple audit reports that can be used for compliance checks or security audits.

Installation

This reporting application is included in the `acitoolkit` package when downloaded using the `git clone` method of installation. The application can be found in the `acitoolkit/applications/reports` directory.

Usage

The application is started from the command line. In its simplest form, it can be invoked by the following command:

```
python aci-report-security-audit.py
```

The full command help is shown below:

```
python aci-report-security-audit.py -h

usage: aci-report-security-audit.py [-h] [-u URL] [-l LOGIN] [-p PASSWORD]
                                    [--csv CSV]

Simple application that logs on to the APIC and produces a report that can be
used for security compliance auditing.

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL     APIC URL e.g. http://1.2.3.4
  -l LOGIN, --login LOGIN
                        APIC login ID.
  -p PASSWORD, --password PASSWORD
                        APIC login password.
  --csv CSV             Output to a CSV file.
```

Output

By default, the audit report is displayed on the screen as comma separated values. If the `--csv` command line option is provided, the output will be sent to the specified filename in proper CSV format.

Each row of the report contains the following information:

```
* Tenant name
* Context (VRF) name
* Bridge Domain name
* Application Profile name
* EPG name
* Number of Consumer EPG Endpoints
* Provided Contract name
* Number of Providing EPG Endpoints
* Consumed Contract name
* Protocol specified in the Filter entry
* Source port range specified in the Filter entry
* Destination port range specified in the Filter entry
```

APIC Test Harness

Purpose

- The APIC Test Harness wraps the Fake APIC into an application server so that toolkit applications can run against the test harness with no modification whatsoever.
- It can be used to execute toolkit scripts and applications against an APIC snapshot rather than a live APIC. For instance, the sample script `acitoolkit/samples/aci-show-endpoints.py` could be run against the live APIC, a snapshot file, yesterday's snapshot file, or even last year's snapshot file.
- It can be used to generate conditions that are very difficult to create in real systems such as communication failures, connection resets, slow responses, and response timeouts.

Usage

1. To generate JSON configuration files, use the snapback application located at `acitoolkit/applications/snapback`
2. Run the `aciconfigdb.py` file

```
python aciconfigdb.py -u <APIC url> -l <login> -p <password> -s --v1 -a
```

   - The `-s` option takes the snapshot of the configuration from the APIC
   - The `--v1` option takes the snapshot using direct HTTP queries rather than the configuration import and export policies. This is important to be able to simulate HTTP responses.
   - The `-a` option ensures the configuration includes all properties of the class objects
     - It's **very important** to give the `-a` because the Fake APIC depends on the all properities of the class objects
   - The JSON files will be located at: `acitoolkit/applications/snapback/apic-config-db`
   - Depending on the APIC, getting all the data may take around 5 - 8 seconds
3. The APIC test harness is located in the `acitoolkit/applications/testharness` directory.

- Run the `apic_test_harness.py` file

```
python apic_test_harness.py --directory <snapshot directory>
```

  - The `--directory` option provides the directory where the snapshot files are located. If the snapshot was created in the `snapback` directory, the command would be issued as follows

```
python apic_test_harness.py --directory ../snapback/apic-config-db/
```

- At this point, the APIC test harness is running as an application server. By default, this service runs on the loopback address `127.0.0.1` on port `5000`.

4. Use the APIC test harness
   - Leave the APIC test harness running and execute applications against it.
   - Here is an example usage taken from the `acitoolkit/samples` directory showing the usage of the `aci-show-endpoints.py`.

```
python aci-show-endpoints.py -l admin -p password -u http://127.0.0.1:5000
```

   - Most of the `show` commands found in the `acitoolkit/samples` directory can be executed against the APIC Test Harness and many applications can be as well.

Full command line options

- The full list of command line arguments is available through the `--help` command line argument.

```
python apic_test_harness.py -h
usage: apic_test_harness.py [-h] [--directory DIRECTORY]
                            [--maxlogfiles MAXLOGFILES]
                            [--debug [{verbose,warnings,critical}]] [--ip␣
↪IP]
                            [--port PORT]
```

```
ACI APIC Test Harness Tool

optional arguments:
  -h, --help            show this help message and exit
  --directory DIRECTORY
                        Directory containing the Snapshot files
  --maxlogfiles MAXLOGFILES
                        Maximum number of log files (default is 10)
  --debug [{verbose,warnings,critical}]
                        Enable debug messages.
  --ip IP               IP address to listen on.
  --port PORT           Port number to listen on.
```

- Log files are stored locally within the directory where the APIC Test Harness is run. For the most complete logs, use the `--debug verbose` command line argument.
- If communication is local only, the default IP address of `127.0.0.1` should be used. If communication will be originated from external sources, the IP address of the interface connecting to the outside world should be used.

What APIC Test Harness supports

- The APIC Test Harness is not a full blown APIC. It can only respond with the information found in the snapshot JSON files. It will accept configuration but the configuration will not change the snapshot JSON files.
- The APIC Test Harness sits on top of the Fake APIC and is limited to what the Fake APIC supports.

Known Issues

- WebSockets and Event Subscriptions are not supported.
- Statistics support is limited.
- No configuration changes are supported.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

c

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W