SoftLayer API Python Client Documentation

Release 5.2.7

SoftLayer Technologies, Inc.

Jul 19, 2017

Contents

API Docs | GitHub | Issues | Pull Requests | PyPI | Twitter | #softlayer on freenode

This is the documentation to SoftLayer's Python API Bindings. These bindings use SoftLayer's XML-RPC interface in order to manage SoftLayer services.

CHAPTER 1

Installation

What's Included

When you install softlayer-python you you will get the following:

- a python package called 'SoftLayer' (casing is important) available in your python path.
- a command-line client placed in your system path named 'slcli'.

Using Pip

Install via pip:

```
$ pip install softlayer
```

Debian/Ubuntu

For Debian "jessie" (currently testing) and Ubuntu 14.04, official system packages are available. **These are typically a couple versions behind so it is recommended to install from pypi if problems are encountered.**

```
$ sudo apt-get install python-softlayer
```

From Source

The project is developed on GitHub, at https://github.com/softlayer/softlayer-python.

Install from source via pip (requires git):

```
$ pip install git+git://github.com/softlayer/softlayer-python.git
```

You can clone the public repository:

```
$ git clone git@github.com:softlayer/softlayer-python.git
```

Or, Download the tarball:

```
$ curl -OL https://github.com/softlayer/softlayer-python/tarball/master
```

Or, download the zipball:

```
$ curl -OL https://github.com/softlayer/softlayer-python/zipball/master
```

Once you have a copy of the source you can install it with one of the following commands:

```
$ python setup.py install
```

Or:

```
$ pip install .
```

For more information about working with the source, or contributing to the project, please see the *Contribution Guide*.

CHAPTER 2

Configuration File

The SoftLayer API bindings load your settings from a number of different locations.

- Input directly into SoftLayer.create_client_from_env(...)
- Enviorment variables (*SL_USERNAME*, *SL_API_KEY*)
- Config file locations (*~/.softlayer*, */etc/softlayer.conf*)
- Or argument (*-C/path/to/config* or *–config=/path/to/config*)

The configuration file is INI-based and requires the *softlayer* section to be present. The only required fields are *username* and *api_key*. You can optionally supply the *endpoint_url* as well. This file is created automatically by the *slcli setup* command detailed here: *Configuration Setup*.

Config Example

```
[softlayer]
username = username
api_key = oyVmeipYQCNrjVS4rF9bHWV7D75S6pa1fghFl384v7mwRCbHTfuJ8qRORIqoVnha
endpoint_url = https://api.softlayer.com/xmlrpc/v3/
timeout = 40
```

CHAPTER 3

API Documentation

This is the primary API client to make API calls. It deals with constructing and executing XML-RPC calls against the SoftLayer API. Below are some links that will help to use the SoftLayer API.

- SoftLayer API Documentation
- Source on GitHub

```
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env(username="username", api_key="api_key")
>>> resp = client.call('Account', 'getObject')
>>> resp['companyName']
'Your Company'
```

Getting Started

You can pass in your username and api_key when creating a SoftLayer client instance. However, you can also set these in the environmental variables 'SL_USERNAME' and 'SL_API_KEY'.

Creating a client instance by passing in the username/api_key:

```
import SoftLayer
client = SoftLayer.create_client_from_env(username='YOUR_USERNAME', api_key='YOUR_API_
↪KEY')
```

Creating a client instance with environmental variables set:

```
$ export SL_USERNAME=YOUR_USERNAME
$ export SL_API_KEY=YOUR_API_KEY
$ python
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env()
```

Below is an example of creating a client instance with more options. This will create a client with the private API endpoint (only accessible from the SoftLayer private network) and a timeout of 4 minutes.

```
client = SoftLayer.create_client_from_env(username='YOUR_USERNAME',
                                          api_key='YOUR_API_KEY'
                                          endpoint_url=SoftLayer.API_PRIVATE_ENDPOINT,
                                          timeout=240)
```

Managers

For day-to-day operation, most users will find the managers to be the most convenient means for interacting with the API. Managers abstract a lot of the complexities of using the API into classes that provide a simpler interface to various services. These are higher-level interfaces to the SoftLayer API.

```
from SoftLayer import VSManager, Client
client = Client(...)
vs = VSManager(client)
vs.list_instances()
[...]
```

Available managers:

SoftLayer.cdn

CDN Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.cdn.**CDNManager**(*client*)
Manage CDN accounts and content.

See product information here: http://www.softlayer.com/content-delivery-network

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_origin(*account_id*, *media_type*, *origin_url*, *cname=None*, *secure=False*)
Adds an original pull mapping to an origin-pull.

Parameters

- **account_id** (*int*) – the numeric ID associated with the CDN account.
- **media_type** (*string*) – the media type/protocol associated with this origin pull mapping; valid values are HTTP, FLASH, and WM.
- **origin_url** (*string*) – the base URL from which content should be pulled.
- **cname** (*string*) – an optional CNAME that should be associated with this origin pull rule; only the hostname should be included (i.e., no 'http://', directories, etc.).
- **secure** (*boolean*) – specifies whether this is an SSL origin pull rule, if SSL is enabled on your account (defaults to false).

get_account(*account_id*, ***kwargs*)
Retrieves a CDN account with the specified account ID.

Parameters

- **int** (*account_id*) – the numeric ID associated with the CDN account.

- ****kwargs** (*dict*) – additional arguments to include in the object mask.

get_origins(*account_id*, ***kwargs*)
Retrieves list of origin pull mappings for a specified CDN account.

Parameters

- **int** (*account_id*) – the numeric ID associated with the CDN account.
- ****kwargs** (*dict*) – additional arguments to include in the object mask.

list_accounts()
Lists CDN accounts for the active user.

load_content(*account_id*, *urls*)
Prefetches one or more URLs to the CDN edge nodes.

Parameters

- **account_id** (*int*) – the CDN account ID into which content should be preloaded.
- **urls** – a string or a list of strings representing the CDN URLs that should be pre-loaded.

Returns true if all load requests were successfully submitted; otherwise, returns the first error encountered.

purge_content(*account_id*, *urls*)
Purges one or more URLs from the CDN edge nodes.

Parameters

- **account_id** (*int*) – the CDN account ID from which content should be purged.
- **urls** – a string or a list of strings representing the CDN URLs that should be purged.

Returns true if all purge requests were successfully submitted; otherwise, returns the first error encountered.

remove_origin(*account_id*, *origin_id*)
Removes an origin pull mapping with the given origin pull ID.

Parameters

- **account_id** (*int*) – the CDN account ID from which the mapping should be deleted.
- **origin_id** (*int*) – the origin pull mapping ID to delete.

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

SoftLayer.dns

DNS Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.dns.**DNSManager**(*client*)
Manage SoftLayer DNS.

See product information here: http://www.softlayer.com/DOMAIN-SERVICES

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

create_record(*zone_id*, *record*, *record_type*, *data*, *ttl=60*)
Create a resource record on a domain.

Parameters

- **id** (*integer*) – the zone's ID
- **record** – the name of the record to add
- **record_type** – the type of record (A, AAAA, CNAME, MX, TXT, etc.)
- **data** – the record's value
- **ttl** (*integer*) – the TTL or time-to-live value (default: 60)

create_zone(*zone*, *serial=None*)
Create a zone for the specified zone.

Parameters

- **zone** – the zone name to create
- **serial** – serial value on the zone (default: strftime(%Y%m%d01))

delete_record(*record_id*)
Delete a resource record by its ID.

Parameters **id** (*integer*) – the record's ID

delete_zone(*zone_id*)
Delete a zone by its ID.

Parameters **zone_id** (*integer*) – the zone ID to delete

dump_zone(*zone_id*)
Retrieve a zone dump in BIND format.

Parameters **id** (*integer*) – The zone ID to dump

edit_record(*record*)
Update an existing record with the options provided.

The provided dict must include an 'id' key and value corresponding to the record that should be updated.

Parameters **record** (*dict*) – the record to update

edit_zone(*zone*)
Update an existing zone with the options provided.

The provided dict must include an 'id' key and value corresponding to the zone that should be updated.

Parameters **zone** (*dict*) – the zone to update

get_record(*record_id*)
Get a DNS record.

Parameters **id** (*integer*) – the record's ID

get_records(*zone_id*, *ttl=None*, *data=None*, *host=None*, *record_type=None*)
List, and optionally filter, records within a zone.

Parameters

- **zone** – the zone name in which to search.
- **ttl** (*int*) – time in seconds

- **data** (*str*) – the records data
- **host** (*str*) – record's host
- **record_type** (*str*) – the type of record

Returns A list of dictionaries representing the matching records within the specified zone.

get_zone(*zone_id*, *records=True*)
Get a zone and its records.

Parameters **zone** – the zone name

Returns A dictionary containing a large amount of information about the specified zone.

list_zones(***kwargs*)
Retrieve a list of all DNS zones.

Parameters ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

Returns A list of dictionaries representing the matching zones.

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

SoftLayer.firewall

Firewall Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.firewall.**FirewallManager**(*client*)
Manages SoftLayer firewalls

See product information here: http://www.softlayer.com/firewalls

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_standard_firewall(*server_id*, *is_virt=True*)
Creates a firewall for the specified virtual/hardware server.

Parameters

- **server_id** (*int*) – The ID of the server to create the firewall for
- **is_virt** (*bool*) – If true, will create the firewall for a virtual server, otherwise for a hardware server.

Returns A dictionary containing the standard virtual server firewall order

add_vlan_firewall(*vlan_id*, *ha_enabled=False*)
Creates a firewall for the specified vlan.

Parameters

- **vlan_id** (*int*) – The ID of the vlan to create the firewall for
- **ha_enabled** (*bool*) – If True, an HA firewall will be created

Returns A dictionary containing the VLAN firewall order

cancel_firewall(*firewall_id*, *dedicated=False*)
Cancels the specified firewall.

Parameters

- **firewall_id** (*int*) – Firewall ID to be cancelled.
- **dedicated** (*bool*) – If true, the firewall instance is dedicated, otherwise, the firewall instance is shared.

edit_dedicated_fwl_rules(*firewall_id*, *rules*)
Edit the rules for dedicated firewall.

Parameters

- **firewall_id** (*integer*) – the instance ID of the dedicated firewall
- **rules** (*list*) – the rules to be pushed on the firewall as defined by SoftLayer_Network_Firewall_Update_Request_Rule

edit_standard_fwl_rules(*firewall_id*, *rules*)
Edit the rules for standard firewall.

Parameters

- **firewall_id** (*integer*) – the instance ID of the standard firewall
- **rules** (*dict*) – the rules to be pushed on the firewall

get_dedicated_fwl_rules(*firewall_id*)
Get the rules of a dedicated firewall.

Parameters **firewall_id** (*integer*) – the instance ID of the dedicated firewall

Returns A list of the rules.

get_dedicated_package(*ha_enabled=False*)
Retrieves the dedicated firewall package.

Parameters **ha_enabled** (*bool*) – True if HA is to be enabled on the firewall False for No HA

Returns A dictionary containing the dedicated virtual server firewall package

get_firewalls()
Returns a list of all firewalls on the account.

Returns A list of firewalls on the current account.

get_standard_fwl_rules(*firewall_id*)
Get the rules of a standard firewall.

Parameters **firewall_id** (*integer*) – the instance ID of the standard firewall

Returns A list of the rules.

get_standard_package(*server_id*, *is_virt=True*)
Retrieves the standard firewall package for the virtual server.

Parameters

- **server_id** (*int*) – The ID of the server to create the firewall for
- **is_virt** (*bool*) – True if the ID provided is for a virtual server, False for a server

Returns A dictionary containing the standard virtual server firewall package

resolve_ids (*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

SoftLayer.managers.firewall.**has_firewall** (*vlan*)
Helper to determine whether or not a VLAN has a firewall.

Parameters **vlan** (*dict*) – A dictionary representing a VLAN

Returns True if the VLAN has a firewall, false if it doesn't.

SoftLayer.hardware

Hardware Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.hardware.**HardwareManager** (*client*, *ordering_manager=None*)
Manage SoftLayer hardware servers.

Example:

```
# Initialize the Manager.
# env variables. These can also be specified in ~/.softlayer,
# or passed directly to SoftLayer.Client()
# SL_USERNAME = YOUR_USERNAME
# SL_API_KEY = YOUR_API_KEY
import SoftLayer
client = SoftLayer.Client()
mgr = SoftLayer.HardwareManager(client)
```

See product information here: http://www.softlayer.com/bare-metal-servers

Parameters

- **client** (*SoftLayer.API.BaseClient*) – the client instance
- **ordering_manager** (*SoftLayer.managers.OrderingManager*) – an optional manager to handle ordering. If none is provided, one will be auto initialized.

cancel_hardware (*hardware_id*, *reason='unneeded'*, *comment=''*, *immediate=False*)
Cancels the specified dedicated server.

Example:

```
# Cancels hardware id 1234
result = mgr.cancel_hardware(hardware_id=1234)
```

Parameters

- **hardware_id** (*int*) – The ID of the hardware to be cancelled.
- **reason** (*string*) – The reason code for the cancellation. This should come from *get_cancellation_reasons()*.
- **comment** (*string*) – An optional comment to include with the cancellation.

change_port_speed(*hardware_id*, *public*, *speed*)
Allows you to change the port speed of a server's NICs.

Parameters

- **hardware_id** (*int*) – The ID of the server
- **public** (*bool*) – Flag to indicate which interface to change. True (default) means the public interface. False indicates the private interface.
- **speed** (*int*) – The port speed to set.

> **Warning:** A port speed of 0 will disable the interface.

Example:

```
#change the Public interface to 10Mbps on instance 12345
result = mgr.change_port_speed(hardware_id=12345,
                               public=True, speed=10)
# result will be True or an Exception
```

edit(*hardware_id*, *userdata=None*, *hostname=None*, *domain=None*, *notes=None*, *tags=None*)
Edit hostname, domain name, notes, user data of the hardware.

Parameters set to None will be ignored and not attempted to be updated.

Parameters

- **hardware_id** (*integer*) – the instance ID to edit
- **userdata** (*string*) – user data on the hardware to edit. If none exist it will be created
- **hostname** (*string*) – valid hostname
- **domain** (*string*) – valid domain name
- **notes** (*string*) – notes about this particular hardware
- **tags** (*string*) – tags to set on the hardware as a comma separated list. Use the empty string to remove all tags.

Example:

```
# Change the hostname on instance 12345 to 'something'
result = mgr.edit(hardware_id=12345 , hostname="something")
#result will be True or an Exception
```

get_cancellation_reasons()
Returns a dictionary of valid cancellation reasons.

These can be used when cancelling a dedicated server via *cancel_hardware()*.

get_create_options()
Returns valid options for ordering hardware.

get_hardware(*hardware_id*, ***kwargs*)
Get details about a hardware device.

Parameters **id** (*integer*) – the hardware ID

Returns A dictionary containing a large amount of information about the specified server.

Example:

```
object_mask = "mask[id,networkVlans[vlanNumber]]"
# Object masks are optional
result = mgr.get_hardware(hardware_id=1234,mask=object_mask)
```

list_hardware(*tags=None*, *cpus=None*, *memory=None*, *hostname=None*, *domain=None*, *datacenter=None*, *nic_speed=None*, *public_ip=None*, *private_ip=None*, ***kwargs*)

List all hardware (servers and bare metal computing instances).

param list tags filter based on tags

param integer cpus filter based on number of CPUS

param integer memory filter based on amount of memory in gigabytes

param string hostname filter based on hostname

param string domain filter based on domain

param string datacenter filter based on datacenter

param integer nic_speed filter based on network speed (in MBPS)

param string public_ip filter based on public ip address

param string private_ip filter based on private ip address

param dict **kwargs response-level options (mask, limit, etc.)

returns Returns a list of dictionaries representing the matching hardware. This list will contain both dedicated servers and bare metal computing instances

Example:

```
# Using a custom object-mask. Will get ONLY what is specified
# These will stem from the SoftLayer_Hardware_Server datatype
object_mask = "mask[hostname,monitoringRobot[robotStatus]]"
result = mgr.list_hardware(mask=object_mask)
```

place_order(***kwargs*)

Places an order for a piece of hardware.

See get_create_options() for valid arguments.

Parameters

- **size** (*string*) – server size name
- **hostname** (*string*) – server hostname
- **domain** (*string*) – server domain name
- **location** (*string*) – location (datacenter) name
- **os** (*string*) – operating system name
- **port_speed** (*int*) – Port speed in Mbps
- **ssh_keys** (*list*) – list of ssh key ids
- **post_uri** (*string*) – The URI of the post-install script to run after reload
- **hourly** (*boolean*) – True if using hourly pricing (default). False for monthly.
- **no_public** (*boolean*) – True if this server should only have private interfaces
- **extras** (*list*) – List of extra feature names

reload(*hardware_id*, *post_uri=None*, *ssh_keys=None*)
Perform an OS reload of a server with its current configuration.

Parameters

- **hardware_id** (*integer*) – the instance ID to reload
- **post_url** (*string*) – The URI of the post-install script to run after reload
- **ssh_keys** (*list*) – The SSH keys to add to the root user

rescue(*hardware_id*)
Reboot a server into the a recsue kernel.

Parameters **instance_id** (*integer*) – the server ID to rescue

Example:

```
result = mgr.rescue(1234)
```

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

update_firmware(*hardware_id*, *ipmi=True*, *raid_controller=True*, *bios=True*, *hard_drive=True*)
Update hardware firmware.

This will cause the server to be unavailable for ~20 minutes.

Parameters

- **hardware_id** (*int*) – The ID of the hardware to have its firmware updated.
- **ipmi** (*bool*) – Update the ipmi firmware.
- **raid_controller** (*bool*) – Update the raid controller firmware.
- **bios** (*bool*) – Update the bios firmware.
- **hard_drive** (*bool*) – Update the hard drive firmware.

Example:

```
# Check the servers active transactions to see progress
result = mgr.update_firmware(hardware_id=1234)
```

verify_order(***kwargs*)
Verifies an order for a piece of hardware.

See *place_order()* for a list of available options.

SoftLayer.image

Image Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.image.**ImageManager**(*client*)
Manages SoftLayer server images.

See product information here: https://knowledgelayer.softlayer.com/topic/image-templates

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

delete_image(*image_id*)
Deletes the specified image.

Parameters **image_id** (*int*) – The ID of the image.

edit(*image_id*, *name=None*, *note=None*, *tag=None*)
Edit image related details.

Parameters

- **image_id** (*int*) – The ID of the image
- **name** (*string*) – Name of the Image.
- **note** (*string*) – Note of the image.
- **tag** (*string*) – Tags of the image to be updated to.

export_image_to_uri(*image_id*, *uri*)
Export image into the given object storage

Parameters

- **image_id** (*int*) – The ID of the image
- **uri** (*string*) – The URI for object storage of the format swift://<objectStorageAccount>@<cluster>/<container>/<objectPath>

get_image(*image_id*, ***kwargs*)
Get details about an image.

Parameters

- **image** (*int*) – The ID of the image.
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

import_image_from_uri(*name*, *uri*, *os_code=None*, *note=None*)
Import a new image from object storage.

Parameters

- **name** (*string*) – Name of the new image
- **uri** (*string*) – The URI for an object storage object (.vhd/.iso file) of the format: swift://<objectStorageAccount>@<cluster>/<container>/<objectPath>
- **os_code** (*string*) – The reference code of the operating system
- **note** (*string*) – Note to add to the image

list_private_images(*guid=None*, *name=None*, ***kwargs*)
List all private images.

Parameters

- **guid** (*string*) – filter based on GUID
- **name** (*string*) – filter based on name
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

list_public_images(*guid=None*, *name=None*, ***kwargs*)
List all public images.

Parameters

- **guid** (*string*) – filter based on GUID
- **name** (*string*) – filter based on name
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

SoftLayer.ipsec

IPSec VPN Manager

license MIT, see LICENSE for more details.

class SoftLayer.managers.ipsec.**IPSECManager**(*client*)
Manage SoftLayer IPSEC VPN tunnel contexts.

This provides helpers to manage IPSEC contexts, private and remote subnets, and NAT translations.

Parameters

- **client** (*SoftLayer.API.BaseClient*) – the client instance
- **account** (*SoftLayer.API.BaseClient*) – account service client
- **context** (*SoftLayer.API.BaseClient*) – tunnel context client
- **customer_subnet** (*SoftLayer.API.BaseClient*) – remote subnet client

add_internal_subnet(*context_id*, *subnet_id*)
Add an internal subnet to a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the internal subnet.

Return bool True if internal subnet addition was successful.

add_remote_subnet(*context_id*, *subnet_id*)
Adds a remote subnet to a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the remote subnet.

Return bool True if remote subnet addition was successful.

add_service_subnet(*context_id*, *subnet_id*)
Adds a service subnet to a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the service subnet.

Return bool True if service subnet addition was successful.

apply_configuration(*context_id*)
Requests network configuration for a tunnel context.

Parameters **context_id** (*int*) – The id-value representing the context instance.

Return bool True if the configuration request was successfully queued.

create_remote_subnet(*account_id*, *identifier*, *cidr*)
Creates a remote subnet on the given account.

Parameters

- **account_id** (*string*) – The account identifier.
- **identifier** (*string*) – The network identifier of the remote subnet.
- **cidr** (*string*) – The CIDR value of the remote subnet.

Return dict Mapping of properties for the new remote subnet.

create_translation(*context_id*, *static_ip*, *remote_ip*, *notes*)
Creates an address translation on a tunnel context/

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **static_ip** (*string*) – The IP address value representing the internal side of the translation entry,
- **remote_ip** (*string*) – The IP address value representing the remote side of the translation entry,
- **notes** (*string*) – The notes to supply with the translation entry,

Return dict Mapping of properties for the new translation entry.

delete_remote_subnet(*subnet_id*)
Deletes a remote subnet from the current account.

Parameters **subnet_id** (*string*) – The id-value representing the remote subnet.

Return bool True if subnet deletion was successful.

get_translation(*context_id*, *translation_id*)
Retrieves a translation entry for the given id values.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **translation_id** (*int*) – The id-value representing the translation instance.

Return dict Mapping of properties for the translation entry.

Raises ***SoftLayerAPIError*** – If a translation cannot be found.

get_translations(*context_id*)
Retrieves all translation entries for a tunnel context.

Parameters **context_id** (*int*) – The id-value representing the context instance.

Return list(dict) Translations associated with the given context

get_tunnel_context(*context_id*, ***kwargs*)
Retrieves the network tunnel context instance.

Parameters **context_id** (*int*) – The id-value representing the context instance.

Return dict Mapping of properties for the tunnel context.

Raises ***SoftLayerAPIError*** – If a context cannot be found.

get_tunnel_contexts(***kwargs*)
Retrieves network tunnel module context instances.

Return list(dict) Contexts associated with the current account.

remove_internal_subnet(*context_id*, *subnet_id*)
Remove an internal subnet from a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the internal subnet.

Return bool True if internal subnet removal was successful.

remove_remote_subnet(*context_id*, *subnet_id*)
Removes a remote subnet from a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the remote subnet.

Return bool True if remote subnet removal was successful.

remove_service_subnet(*context_id*, *subnet_id*)
Removes a service subnet from a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **subnet_id** (*int*) – The id-value representing the service subnet.

Return bool True if service subnet removal was successful.

remove_translation(*context_id*, *translation_id*)
Removes a translation entry from a tunnel context.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **translation_id** (*int*) – The id-value representing the translation.

Return bool True if translation entry removal was successful.

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

update_translation(*context_id*, *translation_id*, *static_ip=None*, *remote_ip=None*, *notes=None*)
Updates an address translation entry using the given values.

Parameters

- **context_id** (*int*) – The id-value representing the context instance.
- **template** (*dict*) – A key-value mapping of translation properties.
- **static_ip** (*string*) – The static IP address value to update.
- **remote_ip** (*string*) – The remote IP address value to update.
- **notes** (*string*) – The notes value to update.

Return bool True if the update was successful.

update_tunnel_context(*context_id*, *friendly_name=None*, *remote_peer=None*, *preshared_key=None*, *phase1_auth=None*, *phase1_crypto=None*, *phase1_dh=None*, *phase1_key_ttl=None*, *phase2_auth=None*, *phase2_crypto=None*, *phase2_dh=None*, *phase2_forward_secrecy=None*, *phase2_key_ttl=None*)

Updates a tunnel context using the given values.

Parameters

- **context_id** (*string*) – The id-value representing the context.
- **friendly_name** (*string*) – The friendly name value to update.
- **remote_peer** (*string*) – The remote peer IP address value to update.
- **preshared_key** (*string*) – The preshared key value to update.
- **phase1_auth** (*string*) – The phase 1 authentication value to update.
- **phase1_crypto** (*string*) – The phase 1 encryption value to update.
- **phase1_dh** (*string*) – The phase 1 diffie hellman group value to update.
- **phase1_key_ttl** (*string*) – The phase 1 key life value to update.
- **phase2_auth** (*string*) – The phase 2 authentication value to update.
- **phase2_crypto** (*string*) – The phase 2 encryption value to update.
- **phase2_df** (*string*) – The phase 2 diffie hellman group value to update.
- **phase2_forward_secriecy** (*string*) – The phase 2 perfect forward secrecy value to update.
- **phase2_key_ttl** (*string*) – The phase 2 key life value to update.

Return bool True if the update was successful.

SoftLayer.load_balancer

Load Balancer Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.load_balancer.**LoadBalancerManager**(*client*)

Manages SoftLayer load balancers.

See product information here: http://www.softlayer.com/load-balancing

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_local_lb(*price_item_id*, *datacenter*)

Creates a local load balancer in the specified data center.

Parameters

- **price_item_id** (*int*) – The price item ID for the load balancer
- **datacenter** (*string*) – The datacenter to create the loadbalancer in

Returns A dictionary containing the product order

add_service(*loadbal_id*, *service_group_id*, *ip_address_id*, *port=80*, *enabled=True*, *hc_type=21*, *weight=1*)
Adds a new service to the service group.

Parameters

- **loadbal_id** (*int*) – The id of the loadbal where the service resides
- **service_group_id** (*int*) – The group to add the service to
- **ip_address id** (*int*) – The ip address ID of the service
- **port** (*int*) – the port of the service
- **enabled** (*bool*) – Enable or disable the service
- **hc_type** (*int*) – The health check type
- **weight** (*int*) – the weight to give to the service

add_service_group(*lb_id*, *allocation=100*, *port=80*, *routing_type=2*, *routing_method=10*)
Adds a new service group to the load balancer.

Parameters

- **loadbal_id** (*int*) – The id of the loadbal where the service resides
- **allocation** (*int*) – percent of connections to allocate toward the group
- **port** (*int*) – the port of the service group
- **routing_type** (*int*) – the routing type to set on the service group
- **routing_method** (*int*) – The routing method to set on the group

cancel_lb(*loadbal_id*)
Cancels the specified load balancer.

Parameters **loadbal_id** (*int*) – Load Balancer ID to be cancelled.

delete_service(*service_id*)
Deletes a service from the loadbal_id.

Parameters **service_id** (*int*) – The id of the service to delete

delete_service_group(*group_id*)
Deletes a service group from the loadbal_id.

Parameters **group_id** (*int*) – The id of the service group to delete

edit_service(*loadbal_id*, *service_id*, *ip_address_id=None*, *port=None*, *enabled=None*, *hc_type=None*, *weight=None*)
Edits an existing service properties.

Parameters

- **loadbal_id** (*int*) – The id of the loadbal where the service resides
- **service_id** (*int*) – The id of the service to edit
- **ip_address** (*string*) – The ip address of the service

- **port** (*int*) – the port of the service
- **enabled** (*bool*) – enable or disable the search
- **hc_type** (*int*) – The health check type
- **weight** (*int*) – the weight to give to the service

edit_service_group(*loadbal_id*, *group_id*, *allocation=None*, *port=None*, *routing_type=None*, *routing_method=None*)

Edit an existing service group.

Parameters

- **loadbal_id** (*int*) – The id of the loadbal where the service resides
- **group_id** (*int*) – The id of the service group
- **allocation** (*int*) – the % of connections to allocate to the group
- **port** (*int*) – the port of the service group
- **routing_type** (*int*) – the routing type to set on the service group
- **routing_method** (*int*) – The routing method to set on the group

get_hc_types()

Retrieves the health check type values.

Returns A dictionary containing the health check types

get_lb_pkgs()

Retrieves the local load balancer packages.

Returns A dictionary containing the load balancer packages

get_local_lb(*loadbal_id*, ***kwargs*)

Returns a specified local load balancer given the id.

Parameters **loadbal_id** (*int*) – The id of the load balancer to retrieve

Returns A dictionary containing the details of the load balancer

get_local_lbs()

Returns a list of all local load balancers on the account.

Returns A list of all local load balancers on the current account.

get_routing_methods()

Retrieves the load balancer routing methods.

Returns A dictionary containing the load balancer routing methods

get_routing_types()

Retrieves the load balancer routing types.

Returns A dictionary containing the load balancer routing types

reset_service_group(*loadbal_id*, *group_id*)

Resets all the connections on the service group.

Parameters

- **loadbal_id** (*int*) – The id of the loadbal
- **group_id** (*int*) – The id of the service group to reset

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

toggle_service_status(*service_id*)
Toggles the service status.

Parameters **service_id** (*int*) – The id of the service to delete

SoftLayer.messaging

Manager for the SoftLayer Message Queue service

license MIT, see LICENSE for more details.

class SoftLayer.managers.messaging.**MessagingConnection**(*account_id*, *endpoint=None*)
Message Queue Service Connection.

Parameters

- **account_id** – Message Queue Account id
- **endpoint** – Endpoint URL

authenticate(*username*, *api_key*, *auth_token=None*)
Authenticate this connection using the given credentials.

Parameters

- **username** – SoftLayer username
- **api_key** – SoftLayer API Key
- **auth_token** – (optional) Starting auth token

create_queue(*queue_name*, ***kwargs*)
Create Queue.

Parameters

- **queue_name** – Queue Name
- ****kwargs** (*dict*) – queue options

create_subscription(*topic_name*, *subscription_type*, ***kwargs*)
Create Subscription.

Parameters

- **topic_name** – Topic Name
- **subscription_type** – type ('queue' or 'http')
- ****kwargs** (*dict*) – Subscription options

create_topic(*topic_name*, ***kwargs*)
Create Topic.

Parameters

- **topic_name** – Topic Name

- ****kwargs** (*dict*) – Topic options

delete_message(*queue_name*, *message_id*)
Delete a message.

Parameters

- **queue_name** – Queue Name
- **message_id** – Message id

delete_queue(*queue_name*, *force=False*)
Delete Queue.

Parameters

- **queue_name** – Queue Name
- **force** – (optional) Force queue to be deleted even if there are pending messages

delete_subscription(*topic_name*, *subscription_id*)
Delete a subscription.

Parameters

- **topic_name** – Topic Name
- **subscription_id** – Subscription id

delete_topic(*topic_name*, *force=False*)
Delete Topic.

Parameters

- **topic_name** – Topic Name
- **force** – (optional) Force topic to be deleted even if there are attached subscribers

get_queue(*queue_name*)
Get queue details.

Parameters **queue_name** – Queue Name

get_queues(*tags=None*)
Get listing of queues.

Parameters **tags** (*list*) – (optional) list of tags to filter by

get_subscriptions(*topic_name*)
Listing of subscriptions on a topic.

Parameters **topic_name** – Topic Name

get_topic(*topic_name*)
Get topic details.

Parameters **topic_name** – Topic Name

get_topics(*tags=None*)
Get listing of topics.

Parameters **tags** (*list*) – (optional) list of tags to filter by

modify_queue(*queue_name*, ***kwargs*)
Modify Queue.

Parameters

- **queue_name** – Queue Name
- ****kwargs** (*dict*) – queue options

modify_topic(*topic_name*, ***kwargs*)
Modify Topic.

Parameters

- **topic_name** – Topic Name
- ****kwargs** (*dict*) – Topic options

pop_message(*queue_name*)
Pop a single message from a queue.

If no messages are returned this returns None

Parameters **queue_name** – Queue Name

pop_messages(*queue_name*, *count=1*)
Pop messages from a queue.

Parameters

- **queue_name** – Queue Name
- **count** – (optional) number of messages to retrieve

push_queue_message(*queue_name*, *body*, ***kwargs*)
Create Queue Message.

Parameters

- **queue_name** – Queue Name
- **body** – Message body
- ****kwargs** (*dict*) – Message options

push_topic_message(*topic_name*, *body*, ***kwargs*)
Create Topic Message.

Parameters

- **topic_name** – Topic Name
- **body** – Message body
- ****kwargs** (*dict*) – Topic message options

stats(*period='hour'*)
Get account stats.

Parameters **period** – 'hour', 'day', 'week', 'month'

class SoftLayer.managers.messaging.**MessagingManager**(*client*)
Manage SoftLayer Message Queue accounts.

See product information here: http://www.softlayer.com/message-queue

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

get_connection(*account_id*, *datacenter=None*, *network=None*)
Get connection to Message Queue Service.

Parameters

- **account_id** – Message Queue Account id

- **datacenter** – Datacenter code
- **network** – network ('public' or 'private')

get_endpoint(*datacenter=None*, *network=None*)
Get a message queue endpoint based on datacenter/network type.

Parameters

- **datacenter** – datacenter code
- **network** – network ('public' or 'private')

get_endpoints()
Get all known message queue endpoints.

list_accounts(***kwargs*)
List message queue accounts.

Parameters ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

ping(*datacenter=None*, *network=None*)
Ping a message queue endpoint.

class SoftLayer.managers.messaging.**QueueAuth**(*endpoint*, *username*, *api_key*, *auth_token=None*)
SoftLayer Message Queue authentication for requests.

Parameters

- **endpoint** – endpoint URL
- **username** – SoftLayer username
- **api_key** – SoftLayer API Key
- **auth_token** – (optional) Starting auth token

auth()
Authenticate.

handle_error(*resp*, **_)
Handle errors.

SoftLayer.metadata

Metadata Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.metadata.**MetadataManager**(*client=None*, *timeout=5*)
Provides an interface for the SoftLayer metadata service.

See product information here: http://sldn.softlayer.com/reference/services/SoftLayer_Resource_Metadata

This provides metadata about the resourse it is called from. See *METADATA_ATTRIBUTES* for full list of attributes.

Usage:

```
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env()
>>> from SoftLayer import MetadataManager
>>> meta = MetadataManager(client)
>>> meta.get('datacenter')
```

```
'dal05'
>>> meta.get('fqdn')
'test.example.com'
```

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

get(*name*, *param=None*)
Retreive a metadata attribute.

Parameters

- **name** (*string*) – name of the attribute to retrieve. See *attribs*
- **param** – Required parameter for some attributes

private_network(***kwargs*)
Returns details about the private network.

Parameters

- **router** (*boolean*) – True to return router details
- **vlans** (*boolean*) – True to return vlan details
- **vlan_ids** (*boolean*) – True to return vlan_ids

public_network(***kwargs*)
Returns details about the public network.

Parameters

- **router** (*boolean*) – True to return router details
- **vlans** (*boolean*) – True to return vlan details
- **vlan_ids** (*boolean*) – True to return vlan_ids

metadata.**METADATA_ATTRIBUTES = ['datacenter', 'domain', 'backend_mac', 'primary_ip', 'primary_backend_ip', 'tags',**

SoftLayer.network

Network Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.network.**NetworkManager**(*client*)
Manage SoftLayer network objects: VLANs, subnets, IPs and rwhois

See product information here: http://www.softlayer.com/networking

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_global_ip(*version=4*, *test_order=False*)
Adds a global IP address to the account.

Parameters

- **version** (*int*) – Specifies whether this is IPv4 or IPv6
- **test_order** (*bool*) – If true, this will only verify the order.

add_subnet(*subnet_type*, *quantity=None*, *vlan_id=None*, *version=4*, *test_order=False*)
Orders a new subnet

Parameters

- **subnet_type** (*str*) – Type of subnet to add: private, public, global
- **quantity** (*int*) – Number of IPs in the subnet
- **vlan_id** (*int*) – VLAN id for the subnet to be placed into
- **version** (*int*) – 4 for IPv4, 6 for IPv6
- **test_order** (*bool*) – If true, this will only verify the order.

assign_global_ip(*global_ip_id*, *target*)
Assigns a global IP address to a specified target.

Parameters

- **global_ip_id** (*int*) – The ID of the global IP being assigned
- **target** (*string*) – The IP address to assign

cancel_global_ip(*global_ip_id*)
Cancels the specified global IP address.

Parameters **id** (*int*) – The ID of the global IP to be cancelled.

cancel_subnet(*subnet_id*)
Cancels the specified subnet.

Parameters **subnet_id** (*int*) – The ID of the subnet to be cancelled.

edit_rwhois(*abuse_email=None*, *address1=None*, *address2=None*, *city=None*, *company_name=None*, *country=None*, *first_name=None*, *last_name=None*, *postal_code=None*, *private_residence=None*, *state=None*)
Edit rwhois record.

get_nas_credentials(*identifier*, ***kwargs*)
Returns a list of IDs of VLANs which match the given VLAN name.

Parameters **instance_id** (*integer*) – the instance ID

Returns A dictionary containing a large amount of information about the specified instance.

get_rwhois()
Returns the RWhois information about the current account.

Returns A dictionary containing the account's RWhois information.

get_subnet(*subnet_id*, ***kwargs*)
Returns information about a single subnet.

Parameters **id** (*string*) – Either the ID for the subnet or its network identifier

Returns A dictionary of information about the subnet

get_vlan(*vlan_id*)
Returns information about a single VLAN.

Parameters **id** (*int*) – The unique identifier for the VLAN

Returns A dictionary containing a large amount of information about the specified VLAN.

ip_lookup(*ip_address*)
Looks up an IP address and returns network information about it.

Parameters **ip_address** (*string*) – An IP address. Can be IPv4 or IPv6

Returns A dictionary of information about the IP

list_global_ips(*version=None*, *identifier=None*, ***kwargs*)
Returns a list of all global IP address records on the account.

Parameters

- **version** (*int*) – Only returns IPs of this version (4 or 6)
- **identifier** (*string*) – If specified, the list will only contain the global ips matching this network identifier.

list_subnets(*identifier=None*, *datacenter=None*, *version=0*, *subnet_type=None*, *network_space=None*, ***kwargs*)
Display a list of all subnets on the account.

This provides a quick overview of all subnets including information about data center residence and the number of devices attached.

Parameters

- **identifier** (*string*) – If specified, the list will only contain the subnet matching this network identifier.
- **datacenter** (*string*) – If specified, the list will only contain subnets in the specified data center.
- **version** (*int*) – Only returns subnets of this version (4 or 6).
- **subnet_type** (*string*) – If specified, it will only returns subnets of this type.
- **network_space** (*string*) – If specified, it will only returns subnets with the given address space label.
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

list_vlans(*datacenter=None*, *vlan_number=None*, *name=None*, ***kwargs*)
Display a list of all VLANs on the account.

This provides a quick overview of all VLANs including information about data center residence and the number of devices attached.

Parameters

- **datacenter** (*string*) – If specified, the list will only contain VLANs in the specified data center.
- **vlan_number** (*int*) – If specified, the list will only contain the VLAN matching this VLAN number.
- **name** (*int*) – If specified, the list will only contain the VLAN matching this VLAN name.
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

resolve_global_ip_ids(*identifier*)
Resolve global ip ids.

resolve_subnet_ids(*identifier*)
Resolve subnet ids.

resolve_vlan_ids(*identifier*)
Resolve VLAN ids.

summary_by_datacenter()
Summary of the networks on the account, grouped by data center.

The resultant dictionary is primarily useful for statistical purposes. It contains count information rather than raw data. If you want raw information, see the `list_vlans()` method instead.

Returns A dictionary keyed by data center with the data containing a set of counts for subnets, hardware, virtual servers, and other objects residing within that data center.

unassign_global_ip(*global_ip_id*)
Unassigns a global IP address from a target.

Parameters **id** (*int*) – The ID of the global IP being unassigned

SoftLayer.sshkey

SSH Key Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.sshkey.**SshKeyManager**(*client*)
Manages account SSH keys in SoftLayer.

See product information here: https://knowledgelayer.softlayer.com/procedure/ssh-keys

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_key(*key*, *label*, *notes=None*)
Adds a new SSH key to the account.

Parameters

- **key** (*string*) – The SSH key to add
- **label** (*string*) – The label for the key

Returns A dictionary of the new key's information.

delete_key(*key_id*)
Permanently deletes an SSH key from the account.

Parameters **key_id** (*int*) – The ID of the key to delete

edit_key(*key_id*, *label=None*, *notes=None*)
Edits information about an SSH key.

Parameters

- **key_id** (*int*) – The ID of the key to edit
- **label** (*string*) – The new label for the key
- **notes** (*string*) – Notes to set or change on the key

Returns A Boolean indicating success or failure

get_key(*key_id*)
Returns full information about a single SSH key.

Parameters **key_id** (*int*) – The ID of the key to retrieve

Returns A dictionary of information about the key

list_keys(*label=None*)
Lists all SSH keys on the account.

Parameters **label** (*string*) – Filter list based on SSH key label

Returns A list of dictionaries with information about each key

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

SoftLayer.ssl

SSL Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.ssl.**SSLManager**(*client*)
Manages SSL certificates in SoftLayer.

See product information here: http://www.softlayer.com/ssl-certificates

Example:

```
# Initialize the Manager.
# env variables. These can also be specified in ~/.softlayer,
# or passed directly to SoftLayer.Client()
# SL_USERNAME = YOUR_USERNAME
# SL_API_KEY = YOUR_API_KEY
import SoftLayer
client = SoftLayer.Client()
mgr = SoftLayer.SSLManager(client)
```

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

add_certificate(*certificate*)
Creates a new certificate.

Parameters **certificate** (*dict*) – A dictionary representing the parts of the certificate. See developer.softlayer.com for more info.

Example:

```
cert = ??
result = mgr.add_certificate(certificate=cert)
```

edit_certificate(*certificate*)
Updates a certificate with the included options.

The provided dict must include an 'id' key and value corresponding to the certificate ID that should be updated.

Parameters **certificate** (*dict*) – the certificate to update.

Example:

```
# Updates the cert id 1234
cert['id'] = 1234
cert['certificate'] = ??
result = mgr.edit_certificate(certificate=cert)
```

get_certificate(*cert_id*)
Gets a certificate with the ID specified.

Parameters **cert_id** (*integer*) – the certificate ID to retrieve

Example:

```
cert = mgr.get_certificate(cert_id=1234)
print(cert)
```

list_certs(*method='all'*)
List all certificates.

Parameters **method** (*string*) – The type of certificates to list. Options are 'all', 'expired', and 'valid'.

Returns A list of dictionaries representing the requested SSL certs.

Example:

```
# Get all valid SSL certs
certs = mgr.list_certs(method='valid')
print certs
```

remove_certificate(*cert_id*)
Removes a certificate.

Parameters **cert_id** (*integer*) – a certificate ID to remove

Example:

```
# Removes certificate with id 1234
result = mgr.remove_certificate(cert_id = 1234)
```

SoftLayer.ticket

Ticket Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.ticket.**TicketManager**(*client*)
Manages SoftLayer support tickets.

See product information here: http://www.softlayer.com/support

Parameters **client** (*SoftLayer.API.BaseClient*) – the client instance

attach_hardware(*ticket_id=None*, *hardware_id=None*)
Attach hardware to a ticket.

Parameters

- **ticket_id** (*integer*) – the id of the ticket to attach to
- **hardware_id** (*integer*) – the id of the hardware to attach

Returns dict – The new ticket attachment

attach_virtual_server(*ticket_id=None*, *virtual_id=None*)
Attach a virtual server to a ticket.

Parameters

- **ticket_id** (*integer*) – the id of the ticket to attach to
- **virtual_id** (*integer*) – the id of the virtual server to attach

Returns dict – The new ticket attachment

create_ticket(*title=None*, *body=None*, *subject=None*)
Create a new ticket.

Parameters

- **title** (*string*) – title for the new ticket
- **body** (*string*) – body for the new ticket
- **subject** (*integer*) – id of the subject to be assigned to the ticket

detach_hardware(*ticket_id=None*, *hardware_id=None*)
Detach hardware from a ticket.

Parameters

- **ticket_id** – the id of the ticket to detach from
- **hardware_id** – the id of the hardware to detach

Returns bool – Whether the detachment was successful

detach_virtual_server(*ticket_id=None*, *virtual_id=None*)
Detach a virtual server from a ticket.

Parameters

- **ticket_id** – the id of the ticket to detach from
- **virtual_id** – the id of the virtual server to detach

Returns bool – Whether the detachment was successful

get_ticket(*ticket_id*)
Get details about a ticket.

Parameters **ticket_id** (*integer*) – the ticket ID

Returns dict – information about the specified ticket

list_subjects()
List all ticket subjects.

list_tickets(*open_status=True*, *closed_status=True*)
List all tickets.

Parameters

- **open_status** (*boolean*) – include open tickets
- **closed_status** (*boolean*) – include closed tickets

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

update_ticket(*ticket_id=None*, *body=None*)
Update a ticket.

Parameters

- **ticket_id** (*integer*) – the id of the ticket to update
- **body** (*string*) – entry to update in the ticket

upload_attachment(*ticket_id=None*, *file_path=None*, *file_name=None*)
Upload an attachment to a ticket.

Parameters

- **ticket_id** (*integer*) – the id of the ticket to upload the attachment to
- **file_path** (*string*) – The path of the attachment to be uploaded
- **file_name** (*string*) – The name of the attachment shown in the ticket

Returns dict – The uploaded attachment

SoftLayer.vs

VS Manager/helpers

license MIT, see LICENSE for more details.

class SoftLayer.managers.vs.**VSManager**(*client*, *ordering_manager=None*)
Manages SoftLayer Virtual Servers.

See product information here: http://www.softlayer.com/virtual-servers

Example:

```
# Initialize the VSManager.
# env variables. These can also be specified in ~/.softlayer,
# or passed directly to SoftLayer.Client()
# SL_USERNAME = YOUR_USERNAME
# SL_API_KEY = YOUR_API_KEY
import SoftLayer
client = SoftLayer.Client()
mgr = SoftLayer.VSManager(client)
```

Parameters

- **client** (*SoftLayer.API.BaseClient*) – the client instance
- **ordering_manager** (*SoftLayer.managers.OrderingManager*) – an optional manager to handle ordering. If none is provided, one will be auto initialized.

cancel_instance(*instance_id*)
Cancel an instance immediately, deleting all its data.

Parameters **instance_id** (*integer*) – the instance ID to cancel

Example:

```
# Cancels instance 12345
mgr.cancel_instance(12345)
```

capture(*instance_id*, *name*, *additional_disks=False*, *notes=None*)
Capture one or all disks from a VS to a SoftLayer image.

Parameters set to None will be ignored and not attempted to be updated.

Parameters

- **instance_id** (*integer*) – the instance ID to edit
- **name** (*string*) – name assigned to the image
- **additional_disks** (*bool*) – set to true to include all additional attached storage devices
- **notes** (*string*) – notes about this particular image

Returns dictionary – information about the capture transaction.

Example:: name = "Testing Images" notes = "Some notes about this image" result = mgr.capture(instance_id=12345, name=name, notes=notes)

change_port_speed(*instance_id*, *public*, *speed*)

Allows you to change the port speed of a virtual server's NICs.

Example:

```
#change the Public interface to 10Mbps on instance 12345
result = mgr.change_port_speed(instance_id=12345,
                               public=True, speed=10)
# result will be True or an Exception
```

Parameters

- **instance_id** (*int*) – The ID of the VS
- **public** (*bool*) – Flag to indicate which interface to change. True (default) means the public interface. False indicates the private interface.
- **speed** (*int*) – The port speed to set.

Warning: A port speed of 0 will disable the interface.

create_instance(***kwargs*)

Creates a new virtual server instance.

Warning: This will add charges to your account

Example:

```
new_vsi = {
    'domain': u'test01.labs.sftlyr.ws',
    'hostname': u'minion05',
    'datacenter': u'hkg02',
    'dedicated': False,
    'private': False,
    'cpus': 1,
    'os_code' : u'UBUNTU_LATEST',
    'hourly': True,
    'ssh_keys': [1234],
    'disks': ('100','25'),
    'local_disk': True,
```

```
    'memory': 1024,
    'tags': 'test, pleaseCancel'
}

vsi = mgr.create_instance(**new_vsi)
# vsi will have the newly created vsi details if done properly.
print vsi
```

Parameters

- **cpus** (*int*) – The number of virtual CPUs to include in the instance.
- **memory** (*int*) – The amount of RAM to order.
- **hourly** (*bool*) – Flag to indicate if this server should be billed hourly (default) or monthly.
- **hostname** (*string*) – The hostname to use for the new server.
- **domain** (*string*) – The domain to use for the new server.
- **local_disk** (*bool*) – Flag to indicate if this should be a local disk (default) or a SAN disk.
- **datacenter** (*string*) – The short name of the data center in which the VS should reside.
- **os_code** (*string*) – The operating system to use. Cannot be specified if image_id is specified.
- **image_id** (*int*) – The ID of the image to load onto the server. Cannot be specified if os_code is specified.
- **dedicated** (*bool*) – Flag to indicate if this should be housed on a dedicated or shared host (default). This will incur a fee on your account.
- **public_vlan** (*int*) – The ID of the public VLAN on which you want this VS placed.
- **private_vlan** (*int*) – The ID of the private VLAN on which you want this VS placed.
- **disks** (*list*) – A list of disk capacities for this server.
- **post_uri** (*string*) – The URI of the post-install script to run after reload
- **private** (*bool*) – If true, the VS will be provisioned only with access to the private network. Defaults to false
- **ssh_keys** (*list*) – The SSH keys to add to the root user
- **nic_speed** (*int*) – The port speed to set
- **tags** (*string*) – tags to set on the VS as a comma separated list

create_instances(*config_list*)

Creates multiple virtual server instances.

This takes a list of dictionaries using the same arguments as create_instance().

> **Warning:** This will add charges to your account

Example:

```
# Define the instance we want to create.
new_vsi = {
    'domain': u'test01.labs.sftlyr.ws',
    'hostname': u'multi-test',
    'datacenter': u'hkg02',
    'dedicated': False,
    'private': False,
    'cpus': 1,
    'os_code' : u'UBUNTU_LATEST',
    'hourly': True,
    'ssh_keys': [87634],
    'disks': ('100','25'),
    'local_disk': True,
    'memory': 1024,
    'tags': 'test, pleaseCancel'
}

# using .copy() so we can make changes to individual nodes
instances = [new_vsi.copy(), new_vsi.copy(), new_vsi.copy()]

# give each its own hostname, not required.
instances[0]['hostname'] = "multi-test01"
instances[1]['hostname'] = "multi-test02"
instances[2]['hostname'] = "multi-test03"

vsi = mgr.create_instances(config_list=instances)
#vsi will be a dictionary of all the new virtual servers
print vsi
```

edit(*instance_id*, *userdata=None*, *hostname=None*, *domain=None*, *notes=None*, *tags=None*)

Edit hostname, domain name, notes, and/or the user data of a VS.

Parameters set to None will be ignored and not attempted to be updated.

Parameters

- **instance_id** (*integer*) – the instance ID to edit
- **userdata** (*string*) – user data on VS to edit. If none exist it will be created
- **hostname** (*string*) – valid hostname
- **domain** (*string*) – valid domain namem
- **notes** (*string*) – notes about this particular VS
- **tags** (*string*) – tags to set on the VS as a comma separated list. Use the empty string to remove all tags.

Returns bool – True or an Exception

Example:: # Change the hostname on instance 12345 to 'something' result = mgr.edit(instance_id=12345 , hostname="something") #result will be True or an Exception

get_create_options()

Retrieves the available options for creating a VS.

Returns A dictionary of creation options.

Example:

```
# Prints out the create option dictionary
options = mgr.get_create_options()
print(options)
```

get_instance(*instance_id*, ***kwargs*)

Get details about a virtual server instance.

Parameters **instance_id** (*integer*) – the instance ID

Returns A dictionary containing a large amount of information about the specified instance.

Example:

```
# Print out instance ID 12345.
vsi = mgr.get_instance(12345)
print vsi

# Print out only FQDN and primaryIP for instance 12345
object_mask = "mask[fullyQualifiedDomainName,primaryIpAddress]"
vsi = mgr.get_instance(12345, mask=mask)
print vsi
```

list_instances(*hourly=True*, *monthly=True*, *tags=None*, *cpus=None*, *memory=None*, *hostname=None*, *domain=None*, *local_disk=None*, *datacenter=None*, *nic_speed=None*, *public_ip=None*, *private_ip=None*, ***kwargs*)

Retrieve a list of all virtual servers on the account.

Example:

```
# Print out a list of hourly instances in the DAL05 data center.

for vsi in mgr.list_instances(hourly=True, datacenter='dal05'):
   print vsi['fullyQualifiedDomainName'], vsi['primaryIpAddress']

# Using a custom object-mask. Will get ONLY what is specified
object_mask = "mask[hostname,monitoringRobot[robotStatus]]"
for vsi in mgr.list_instances(mask=object_mask,hourly=True):
    print vsi
```

Parameters

- **hourly** (*boolean*) – include hourly instances
- **monthly** (*boolean*) – include monthly instances
- **tags** (*list*) – filter based on list of tags
- **cpus** (*integer*) – filter based on number of CPUS
- **memory** (*integer*) – filter based on amount of memory
- **hostname** (*string*) – filter based on hostname
- **domain** (*string*) – filter based on domain
- **local_disk** (*string*) – filter based on local_disk
- **datacenter** (*string*) – filter based on datacenter
- **nic_speed** (*integer*) – filter based on network speed (in MBPS)
- **public_ip** (*string*) – filter based on public ip address

- **private_ip** (*string*) – filter based on private ip address
- ****kwargs** (*dict*) – response-level options (mask, limit, etc.)

Returns Returns a list of dictionaries representing the matching virtual servers

reload_instance(*instance_id*, *post_uri=None*, *ssh_keys=None*, *image_id=None*)
Perform an OS reload of an instance.

Parameters

- **instance_id** (*integer*) – the instance ID to reload
- **post_url** (*string*) – The URI of the post-install script to run after reload
- **ssh_keys** (*list*) – The SSH keys to add to the root user
- **image_id** (*int*) – The ID of the image to load onto the server

Warning: This will reformat the primary drive. Post-provision script MUST be HTTPS for it to be executed.

Example:

```
# Reload instance ID 12345 then run a custom post-provision script.
# Post-provision script MUST be HTTPS for it to be executed.
post_uri = 'https://somehost.com/bootstrap.sh'
vsi = mgr.reload_instance(12345, post_uri=post_url)
```

rescue(*instance_id*)
Reboot a VSI into the Xen recsue kernel.

Parameters **instance_id** (*integer*) – the instance ID to rescue

Returns bool – True or an Exception

Example:: # Puts instance 12345 into rescue mode result = mgr.rescue(instance_id=12345)

resolve_ids(*identifier*)
Takes a string and tries to resolve to a list of matching ids.

What exactly 'identifier' can be depends on the resolvers

Parameters **identifier** (*string*) – identifying string

Returns list

upgrade(*instance_id*, *cpus=None*, *memory=None*, *nic_speed=None*, *public=True*)
Upgrades a VS instance.

Example:

```
# Upgrade instance 12345 to 4 CPUs and 4 GB of memory
import SoftLayer
client = SoftLayer.create_client_from_env()
mgr = SoftLayer.VSManager(client)
mgr.upgrade(12345, cpus=4, memory=4)
```

Parameters

- **instance_id** (*int*) – Instance id of the VS to be upgraded

- **cpus** (*int*) – The number of virtual CPUs to upgrade to of a VS instance.
- **public** (*bool*) – CPU will be in Private/Public Node.
- **memory** (*int*) – RAM of the VS to be upgraded to.
- **nic_speed** (*int*) – The port speed to set

Returns bool

verify_create_instance(***kwargs*)

Verifies an instance creation command.

Without actually placing an order. See *create_instance()* for a list of available options.

Example:

```
new_vsi = {
    'domain': u'test01.labs.sftlyr.ws',
    'hostname': u'minion05',
    'datacenter': u'hkg02',
    'dedicated': False,
    'private': False,
    'cpus': 1,
    'os_code' : u'UBUNTU_LATEST',
    'hourly': True,
    'ssh_keys': [1234],
    'disks': ('100','25'),
    'local_disk': True,
    'memory': 1024
}

vsi = mgr.verify_create_instance(**new_vsi)
# vsi will be a SoftLayer_Container_Product_Order_Virtual_Guest
# if your order is correct. Otherwise you will get an exception
print vsi
```

wait_for_ready(*instance_id*, *limit*, *delay=1*, *pending=False*)

Determine if a VS is ready and available.

In some cases though, that can mean that no transactions are running. The default arguments imply a VS is operational and ready for use by having network connectivity and remote access is available. Setting `pending=True` will ensure future API calls against this instance will not error due to pending transactions such as OS Reloads and cancellations.

Parameters

- **instance_id** (*int*) – The instance ID with the pending transaction
- **limit** (*int*) – The maximum amount of time to wait.
- **delay** (*int*) – The number of seconds to sleep before checks. Defaults to 1.
- **pending** (*bool*) – Wait for pending transactions not related to provisioning or reloads such as monitoring.

Example:

```
# Will return once vsi 12345 is ready, or after 10 checks
ready = mgr.wait_for_ready(12345, 10)
```

wait_for_transaction(*instance_id*, *limit*, *delay=1*)

Waits on a VS transaction for the specified amount of time.

This is really just a wrapper for wait_for_ready(pending=True). Provided for backwards compatibility.

Parameters

- **instance_id** (*int*) – The instance ID with the pending transaction
- **limit** (*int*) – The maximum amount of time to wait.
- **delay** (*int*) – The number of seconds to sleep before checks. Defaults to 1.

If you need more power or functionality than the managers provide, you can make direct API calls as well.

Making API Calls

For full control over your account and services, you can directly call the SoftLayer API. The SoftLayer API client for python leverages SoftLayer's XML-RPC API. It supports authentication, object masks, object filters, limits, offsets, and retrieving objects by id. The following section assumes you have an initialized client named 'client'.

The best way to test our setup is to call the getObject method on the SoftLayer_Account service.

```
client.call('Account', 'getObject')
```

For a more complex example we'll retrieve a support ticket with id 123456 along with the ticket's updates, the user it's assigned to, the servers attached to it, and the datacenter those servers are in. To retrieve our extra information using an object mask.

Retrieve a ticket using object masks.

```
ticket = client.call('Ticket', 'getObject',
    id=123456, mask="updates, assignedUser, attachedHardware.datacenter")
```

Now add an update to the ticket with Ticket.addUpdate. This uses a parameter, which translate to positional arguments in the order that they appear in the API docs.

```
update = client.call('Ticket', 'addUpdate', {'entry' : 'Hello!'}, id=123456)
```

Let's get a listing of virtual guests using the domain example.com

```
client.call('Account', 'getVirtualGuests',
    filter={'virtualGuests': {'domain': {'operation': 'example.com'}}})
```

This call gets tickets created between the beginning of March 1, 2013 and March 15, 2013.

```
client.call('Account', 'getTickets',
    filter={
        'tickets': {
            'createDate': {
                'operation': 'betweenDate',
                'options': [
                    {'name': 'startDate', 'value': ['03/01/2013 0:0:0']},
                    {'name': 'endDate', 'value': ['03/15/2013 23:59:59']}
                ]
            }
        }
    }
)
```

SoftLayer's XML-RPC API also allows for pagination.

```
client.call('Account', 'getVirtualGuests', limit=10, offset=0)  # Page 1
client.call('Account', 'getVirtualGuests', limit=10, offset=10)  # Page 2
```

Here's how to create a new Cloud Compute Instance using SoftLayer_Virtual_Guest.createObject. Be warned, this call actually creates an hourly virtual server so this will have billing implications.

```
client.call('Virtual_Guest', 'createObject', {
        'hostname': 'myhostname',
        'domain': 'example.com',
        'startCpus': 1,
        'maxMemory': 1024,
        'hourlyBillingFlag': 'true',
        'operatingSystemReferenceCode': 'UBUNTU_LATEST',
        'localDiskFlag': 'false'
    })
```

API Reference

SoftLayer Python API Client

SoftLayer API bindings

Usage:

```
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env(username="username",
                                              api_key="api_key")
>>> resp = client.call('Account', 'getObject')
>>> resp['companyName']
'Your Company'
```

license MIT, see LICENSE for more details.

class SoftLayer.**BaseClient**(*auth=None*, *transport=None*)

Base SoftLayer API client.

Parameters

- **auth** – auth driver that looks like SoftLayer.auth.AuthenticationBase
- **transport** – An object that's callable with this signature: transport(SoftLayer.transports.Request)

authenticate_with_password(*username*, *password*, *security_question_id=None*, *security_question_answer=None*)

Performs Username/Password Authentication

Parameters

- **username** (*string*) – your SoftLayer username
- **password** (*string*) – your SoftLayer password
- **security_question_id** (*int*) – The security question id to answer
- **security_question_answer** (*string*) – The answer to the security question

call(*service*, *method*, **args*, ***kwargs*)

Make a SoftLayer API call.

Parameters

- **method** – the method to call on the service
- ***args** – (optional) arguments for the remote call
- **id** – (optional) id for the resource
- **mask** – (optional) object mask
- **filter** (*dict*) – (optional) filter dict
- **headers** (*dict*) – (optional) optional XML-RPC headers
- **compress** (*boolean*) – (optional) Enable/Disable HTTP compression
- **raw_headers** (*dict*) – (optional) HTTP transport headers
- **limit** (*int*) – (optional) return at most this many results
- **offset** (*int*) – (optional) offset results by this many
- **iter** (*boolean*) – (optional) if True, returns a generator with the results
- **verify** (*bool*) – verify SSL cert
- **cert** – client certificate path

Usage:

```
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env()
>>> client.call('Account', 'getVirtualGuests', mask="id", limit=10)
[...]
```

iter_call(*service*, *method*, **args*, ***kwargs*)

A generator that deals with paginating through results.

Parameters

- **service** – the name of the SoftLayer API service
- **method** – the method to call on the service
- **chunk** (*integer*) – result size for each API call (defaults to 100)
- ***args** – same optional arguments that `Service.call` takes
- ****kwargs** – same optional keyword arguments that `Service.call` takes

`SoftLayer.`**create_client_from_env**(*username=None*, *api_key=None*, *endpoint_url=None*, *timeout=None*, *auth=None*, *config_file=None*, *proxy=None*, *user_agent=None*, *transport=None*, *verify=True*)

Creates a SoftLayer API client using your environment.

Settings are loaded via keyword arguments, environemtal variables and config file.

Parameters

- **username** – an optional API username if you wish to bypass the package's built-in username
- **api_key** – an optional API key if you wish to bypass the package's built in API key
- **endpoint_url** – the API endpoint base URL you wish to connect to. Set this to API_PRIVATE_ENDPOINT to connect via SoftLayer's private network.

- **proxy** – proxy to be used to make API calls
- **timeout** (*integer*) – timeout for API requests
- **auth** – an object which responds to get_headers() to be inserted into the xml-rpc headers. Example: *BasicAuthentication*
- **config_file** – A path to a configuration file used to load settings
- **user_agent** – an optional User Agent to report when making API calls if you wish to bypass the packages built in User Agent string
- **transport** – An object that's callable with this signature: transport(SoftLayer.transports.Request)
- **verify** (*bool*) – decide to verify the server's SSL/TLS cert. DO NOT SET TO FALSE WITHOUT UNDERSTANDING THE IMPLICATIONS.

Usage:

```
>>> import SoftLayer
>>> client = SoftLayer.create_client_from_env()
>>> resp = client.call('Account', 'getObject')
>>> resp['companyName']
'Your Company'
```

SoftLayer.**Client**(***kwargs*)
Get a SoftLayer API Client using environmental settings.

Deprecated in favor of create_client_from_env()

class SoftLayer.**BasicAuthentication**(*username*, *api_key*)
Token-based authentication class.

Parameters

- **str** (*api_key*) – a user's username
- **str** – a user's API key

get_request(*request*)
Sets token-based auth headers.

exception SoftLayer.**SoftLayerError**
The base SoftLayer error.

exception SoftLayer.**SoftLayerAPIError**(*fault_code*, *fault_string*, **args*)
SoftLayerAPIError is an exception raised during API errors.

Provides faultCode and faultString properties.

class SoftLayer.**SoftLayerListResult**(*items*, *total_count*)
A SoftLayer API list result.

CHAPTER 4

Command-line Interface

The SoftLayer command line interface is available via the *slcli* command available in your *PATH*. The *slcli* command is a reference implementation of SoftLayer API bindings for python and how to efficiently make API calls. See the *Usage Examples* section to see how to discover all of the functionality not fully documented here.

Interacting with IPSEC Tunnels

The IPSEC *Command-line Interface* commands can be used to configure an existing IPSEC tunnel context. Subnets in the SoftLayer private network can be associated to the tunnel context along with user-defined remote subnets. Address translation entries may also be defined to provide NAT functionality from static subnet IP addresses associated with the tunnel context to user-defined remote subnet IP addresses.

Note: Most CLI actions that affect an IPSEC tunnel context do not result in configuration changes to SoftLayer network devices. A separate *configure* command is available to issue a device configuration request.

To see more information about the IPSEC tunnel context module and API interaction, see *IPSEC Module* documentation.

ipsec list

A list of all IPSEC tunnel contexts associated with the current user's account can be retrieved via the `ipsec list` command. This provides a brief overview of all tunnel contexts and can be used to retrieve an individual context's identifier, which all other CLI commands require.

```
$ slcli ipsec list
:.....:.........:.............:........................:.....................
↪:........................:
:  id :   name   : friendly name : internal peer IP address : remote peer IP address␣
↪:          created          :
:.....:.........:.............:........................:.....................
↪:........................:
```

```
: 445 : ipsec038 :  ipsec tunnel :      173.192.250.79      :      158.85.80.22
↪: 2012-03-05T14:07:34-06:00 :
:.....:.........:..............:..........................:.....................
↪:........................:
```

ipsec detail

More detailed information can be retrieved for an individual context using the `ipsec detail` command. Using the detail command, information about associated internal subnets, remote subnets, static subnets, service subnets and address translations may also be retrieved using multiple instances of the `-i|--include` option.

```
$ slcli ipsec detail 445 -i at -i is -i rs -i sr -i ss
Context Details:
:................................:........................:
:                           name : value                  :
:................................:........................:
:                             id : 445                    :
:                           name : ipsec038               :
:                  friendly name : ipsec tunnel           :
:        internal peer IP address : 173.192.250.79        :
:          remote peer IP address : 158.85.80.22          :
:     advanced configuration flag : 0                     :
:                  preshared key : secret                 :
:          phase 1 authentication : MD5                    :
:    phase 1 diffie hellman group : 0                      :
:              phase 1 encryption : DES                    :
:                phase 1 key life : 240                    :
:          phase 2 authentication : MD5                    :
:    phase 2 diffie hellman group : 1                      :
:              phase 2 encryption : DES                    :
:                phase 2 key life : 240                    :
: phase 2 perfect forward secrecy : 1                      :
:                         created : 2012-03-05T14:07:34-06:00 :
:                        modified : 2017-05-17T12:01:33-06:00 :
:................................:........................:
Address Translations:
:.......:................:...................:................:.............
↪........:..............:
:   id  : static IP address : static IP address id : remote IP address : remote IP
↪address id :       note      :
:.......:................:...................:................:.............
↪........:..............:
: 15920 :    10.1.249.86    :       9791681       :    158.85.80.22   :        98828
↪        :  windows server :
: 15918 :    10.1.249.84    :       9791679       :    158.85.80.20   :        98824
↪        :   unix server   :
:.......:................:...................:................:.............
↪........:..............:
Internal Subnets:
:........:................:......:......:
:   id   : network identifier : cidr : note :
:........:................:......:......:
: 180767 :    10.28.67.128    :  26  :      :
:........:................:......:......:
Remote Subnets:
:......:................:......:......:
```

```
:  id  : network identifier : cidr : note :
:......:..................:......:......:
: 7852 :    158.85.80.20    :  30  :      :
:......:..................:......:......:
Static Subnets:
:........:..................:......:......:
:   id   : network identifier : cidr : note :
:........:..................:......:......:
: 231807 :    10.1.249.84     :  30  :      :
:........:..................:......:......:
Service Subnets:
:........:..................:......:......:
:   id   : network identifier : cidr : note :
:........:..................:......:......:
: 162079 :     10.0.80.0      :  25  :      :
:........:..................:......:......:
```

ipsec update

Most values listed in the tunnel context detail printout can be modified using the `ipsec update` command. The following is given when executing with the `-h|--help` option and highlights all properties that may be modified.

```
$ slcli ipsec update -h
Usage: slcli ipsec update [OPTIONS] CONTEXT_ID

  Update tunnel context properties.

  Updates are made atomically, so either all are accepted or none are.

  Key life values must be in the range 120-172800.

  Phase 2 perfect forward secrecy must be in the range 0-1.

  A separate configuration request should be made to realize changes on
  network devices.

Options:
  --friendly-name TEXT            Friendly name value
  --remote-peer TEXT              Remote peer IP address value
  --preshared-key TEXT            Preshared key value
  --p1-auth, --phase1-auth [MD5|SHA1|SHA256]
                                  Phase 1 authentication value
  --p1-crypto, --phase1-crypto [DES|3DES|AES128|AES192|AES256]
                                  Phase 1 encryption value
  --p1-dh, --phase1-dh [0|1|2|5]  Phase 1 diffie hellman group value
  --p1-key-ttl, --phase1-key-ttl INTEGER RANGE
                                  Phase 1 key life value
  --p2-auth, --phase2-auth [MD5|SHA1|SHA256]
                                  Phase 2 authentication value
  --p2-crypto, --phase2-crypto [DES|3DES|AES128|AES192|AES256]
                                  Phase 2 encryption value
  --p2-dh, --phase2-dh [0|1|2|5]  Phase 2 diffie hellman group value
  --p2-forward-secrecy, --phase2-forward-secrecy INTEGER RANGE
                                  Phase 2 perfect forward secrecy value
  --p2-key-ttl, --phase2-key-ttl INTEGER RANGE
                                  Phase 2 key life value
```

```
  -h, --help                    Show this message and exit.
```

ipsec configure

A request to configure SoftLayer network devices for a given tunnel context can be issued using the `ipsec configure` command.

Note: Once a configuration request is received, the IPSEC tunnel context will be placed into an unmodifiable state, and further changes against the tunnel context will be prevented. Once configuration changes have been made, the tunnel context may again be modified. The unmodifiable state of a tunnel context is indicated by an *advanced configuration flag* value of 1.

ipsec subnet-add

Internal, remote and service subnets can be associated to an IPSEC tunnel context using the `ipsec subnet-add` command. Additionally, remote subnets can be created using this same command, which will then be associated to the targeted tunnel context.

Note: The targeted subnet type must be specified. A subnet id must be provided when associating internal and service subnets. Either a subnet id or a network identifier must be provided when associating remote subnets. If a network identifier is provided when associating a remote subnet, that subnet will first be created and then associated to the tunnel context.

The following is an exmaple of associating an internal subnet to a tunnel context.

```
$ slcli ipsec subnet-add 445 --subnet-id 180767 --subnet-type internal
Added internal subnet #180767
```

The following is an example of creating and associating a remote subnet to a tunnel context.

```
$ slcli ipsec subnet-add 445 --subnet-type remote --network 50.100.0.0/26
Created subnet 50.100.0.0/26 #21268
Added remote subnet #21268
```

ipsec subnet-remove

Internal, remote and service subnets can be disassociated from an IPSEC tunnel context via the `ipsec subnet-remove` command.

Note: The targeted subnet id and type must be specified. When disassociating remote subnets, that subnet record will also be deleted.

The following is an example of disassociating an internal subnet from a tunnel context.

```
$ slcli ipsec subnet-remove 445 --subnet-id 180767 --subnet-type internal
Removed internal subnet #180767
```

ipsec translation-add

Address translation entries can be added to a tunnel context to provide NAT functionality from a statically routed subnet associated with the tunnel context to a remote subnet. This action is performed with the `ipsec translation-add` command.

Note: Both static and remote IP address values must be specified. An optional note value may also be provided.

The following is an example of adding a new address translation entry.

```
$ slcli ipsec translation-add 445 --static-ip 10.1.249.87 --remote-ip 50.100.0.10 --
↪note 'email server'
Created translation from 10.1.249.87 to 50.100.0.10 #15922
```

ipsec translation-remove

Address translation entries can be removed using the `ipsec translation-remove` command.

The following is an example of removing an address translation entry.

```
$ slcli ipsec translation-remove 445 --translation-id 15922
Removed translation #15922
```

ipsec translation-update

Address translation entries may also be modified using the `ipsec translation-update` command.

The following is an example of updating an existing address translation entry.

```
$ slcli ipsec translation-update 445 --translation-id 15924 --static-ip 10.1.249.86 --
↪remote-ip 50.100.0.8 --note 'new email server'
Updated translation #15924
```

Working with Virtual Servers

Using the SoftLayer portal to order virtual servers is fine, but for a number of reasons it's often more convenient to use the command line. For this, you can use SoftLayer's command-line client to make administrative tasks quicker and easier. This page gives an intro to working with SoftLayer virtual servers using SoftLayer's command-line client.

Note: The following assumes that the client is already *configured with valid SoftLayer credentials*.

First, let's list the current virtual servers with *slcli vs list*.

```
$ slcli vs list
:.....:...........:........................:......:.......:............:.......
↪.....:.................:........:
:  id : datacenter :           host          : cores : memory :  primary_ip  : ␣
↪backend_ip : active_transaction : owner  :
```

```
:.....:...........:........................:......:........:.............:........
↪.....:..................:........:
:.....:...........:........................:......:........:.............:........
↪.....:..................:........:
```

We don't have any virtual servers yet! Let's fix that. Before we can create a virtual server (VS), we need to know what options are available to us: RAM, CPU, operating systems, disk sizes, disk types, datacenters, and so on. Luckily, there's a simple command to show all options: *slcli vs create-options*.

```
$ slcli vs create-options
:................:.....................................................................
↪.......................:
:            Name : Value                                                           ␣
↪                        :
:................:.....................................................................
↪.......................:
:      datacenter : ams01,dal01,dal05,dal06,dal09,hkg02,hou02,lon02,mel01,par01,sea01,
↪sjc01,sng01,tor01,wdc01 :
:  cpus (private) : 1,2,4,8                                                           ␣
↪                        :
: cpus (standard) : 1,2,4,8,12,16                                                     ␣
↪                        :
:          memory : 1024,2048,4096,6144,8192,12288,16384,32768,49152,65536            ␣
↪                        :
:     os (CENTOS) : CENTOS_5_32                                                       ␣
↪                        :
:                 : CENTOS_5_64                                                       ␣
↪                        :
:                 : CENTOS_6_32                                                       ␣
↪                        :
:                 : CENTOS_6_64                                                       ␣
↪                        :
:                 : CENTOS_7_64                                                       ␣
↪                        :
:                 : CENTOS_LATEST                                                     ␣
↪                        :
:                 : CENTOS_LATEST_32                                                  ␣
↪                        :
:                 : CENTOS_LATEST_64                                                  ␣
↪                        :
: os (CLOUDLINUX) : CLOUDLINUX_5_32                                                   ␣
↪                        :
:                 : CLOUDLINUX_5_64                                                   ␣
↪                        :
:                 : CLOUDLINUX_6_32                                                   ␣
↪                        :
:                 : CLOUDLINUX_6_64                                                   ␣
↪                        :
:                 : CLOUDLINUX_LATEST                                                 ␣
↪                        :
:                 : CLOUDLINUX_LATEST_32                                              ␣
↪                        :
:                 : CLOUDLINUX_LATEST_64                                              ␣
↪                        :
:     os (DEBIAN) : DEBIAN_6_32                                                       ␣
↪                        :
:                 : DEBIAN_6_64                                                       ␣
↪                        :
```

```
:                  : DEBIAN_7_32                                                ␣
↪                         :
:                  : DEBIAN_7_64                                                ␣
↪                         :
:                  : DEBIAN_LATEST                                              ␣
↪                         :
:                  : DEBIAN_LATEST_32                                           ␣
↪                         :
:                  : DEBIAN_LATEST_64                                           ␣
↪                         :
:     os (REDHAT)  : REDHAT_5_32                                                ␣
↪                         :
:                  : REDHAT_5_64                                                ␣
↪                         :
:                  : REDHAT_6_32                                                ␣
↪                         :
:                  : REDHAT_6_64                                                ␣
↪                         :
:                  : REDHAT_LATEST                                              ␣
↪                         :
:                  : REDHAT_LATEST_32                                           ␣
↪                         :
:                  : REDHAT_LATEST_64                                           ␣
↪                         :
:     os (UBUNTU)  : UBUNTU_10_32                                               ␣
↪                         :
:                  : UBUNTU_10_64                                               ␣
↪                         :
:                  : UBUNTU_12_32                                               ␣
↪                         :
:                  : UBUNTU_12_64                                               ␣
↪                         :
:                  : UBUNTU_14_32                                               ␣
↪                         :
:                  : UBUNTU_14_64                                               ␣
↪                         :
:                  : UBUNTU_LATEST                                              ␣
↪                         :
:                  : UBUNTU_LATEST_32                                           ␣
↪                         :
:                  : UBUNTU_LATEST_64                                           ␣
↪                         :
:   os (VYATTACE)  : VYATTACE_6.5_64                                            ␣
↪                         :
:                  : VYATTACE_6.6_64                                            ␣
↪                         :
:                  : VYATTACE_LATEST                                            ␣
↪                         :
:                  : VYATTACE_LATEST_64                                         ␣
↪                         :
:        os (WIN)  : WIN_2003-DC-SP2-1_32                                       ␣
↪                         :
:                  : WIN_2003-DC-SP2-1_64                                       ␣
↪                         :
:                  : WIN_2003-ENT-SP2-5_32                                      ␣
↪                         :
:                  : WIN_2003-ENT-SP2-5_64                                      ␣
↪                         :
```

```
:                  : WIN_2003-STD-SP2-5_32                                                 ␣
↪                        :
:                  : WIN_2003-STD-SP2-5_64                                                 ␣
↪                        :
:                  : WIN_2008-DC-R2_64                                                     ␣
↪                        :
:                  : WIN_2008-DC-SP2_64                                                    ␣
↪                        :
:                  : WIN_2008-ENT-R2_64                                                    ␣
↪                        :
:                  : WIN_2008-ENT-SP2_32                                                   ␣
↪                        :
:                  : WIN_2008-ENT-SP2_64                                                   ␣
↪                        :
:                  : WIN_2008-STD-R2-SP1_64                                                ␣
↪                        :
:                  : WIN_2008-STD-R2_64                                                    ␣
↪                        :
:                  : WIN_2008-STD-SP2_32                                                   ␣
↪                        :
:                  : WIN_2008-STD-SP2_64                                                   ␣
↪                        :
:                  : WIN_2012-DC_64                                                        ␣
↪                        :
:                  : WIN_2012-STD_64                                                       ␣
↪                        :
:                  : WIN_LATEST                                                            ␣
↪                        :
:                  : WIN_LATEST_32                                                         ␣
↪                        :
:                  : WIN_LATEST_64                                                         ␣
↪                        :
:   local disk(0)  : 25,100                                                                ␣
↪                        :
:   local disk(2)  : 25,100,150,200,300                                                    ␣
↪                        :
:     san disk(0)  : 25,100                                                                ␣
↪                        :
:     san disk(2)  : 10,20,25,30,40,50,75,100,125,150,175,200,250,300,350,400,500,750,
↪1000,1500,2000          :
:     san disk(3)  : 10,20,25,30,40,50,75,100,125,150,175,200,250,300,350,400,500,750,
↪1000,1500,2000          :
:     san disk(4)  : 10,20,25,30,40,50,75,100,125,150,175,200,250,300,350,400,500,750,
↪1000,1500,2000          :
:     san disk(5)  : 10,20,25,30,40,50,75,100,125,150,175,200,250,300,350,400,500,750,
↪1000,1500,2000          :
:             nic  : 10,100,1000                                                         ␣
↪                        :
:................:.....................................................................
↪.....................:
```

Here's the command to create a 2-core virtual server with 1GiB memory, running Ubuntu 14.04 LTS, and that is billed on an hourly basis in the San Jose 1 datacenter using the command *slcli vs create*.

```
$ slcli vs create --hostname=example --domain=softlayer.com --cpu 2 --memory 1024 -o␣
↪UBUNTU_14_64 --datacenter=sjc01 --billing=hourly
This action will incur charges on your account. Continue? [y/N]: y
:.........:....................................:
```

```
:    name : value                                :
:.........:........................................:
:      id : 1234567                              :
: created : 2013-06-13T08:29:44-06:00            :
:    guid : 6e013cde-a863-46ee-8s9a-f806dba97c89 :
:.........:........................................:
```

After the last command, the virtual server is now being built. It should instantly appear in your virtual server list now.

```
$ slcli vs list
:.........:...........:........................:......:........:................:....
↪.........:.................:
:    id   : datacenter :          host          : cores : memory :   primary_ip   : ␣
↪backend_ip  : active_transaction :
:.........:...........:........................:......:........:................:....
↪.........:.................:
: 1234567 :   sjc01    : example.softlayer.com :   2   :   1G   : 108.168.200.11 : 10.
↪54.80.200 :    Assign Host     :
:.........:...........:........................:......:........:................:....
↪.........:.................:
```

Cool. You may ask, "It's creating... but how do I know when it's done?" Well, here's how:

```
$ slcli vs ready 'example' --wait=600
READY
```

When the previous command returns, you'll know that the virtual server has finished the provisioning process and is ready to use. This is *very* useful for chaining commands together.

Now that you have your virtual server, let's get access to it. To do that, use the *slcli vs detail* command. From the example below, you can see that the username is 'root' and password is 'ABCDEFGH'.

Warning: Be careful when using the *–passwords* flag. This will print the virtual server's password on the screen. Make sure no one is looking over your shoulder. It's also advisable to change your root password soon after creating your virtual server, or to create a user with sudo access and disable SSH-based login directly to the root account.

```
$ slcli vs detail example --passwords
:..............:..............................:
:         Name : Value                        :
:..............:..............................:
:           id : 1234567                      :
:     hostname : example.softlayer.com        :
:       status : Active                       :
:        state : Running                      :
:   datacenter : sjc01                        :
:        cores : 2                            :
:       memory : 1G                           :
:    public_ip : 108.168.200.11               :
:   private_ip : 10.54.80.200                 :
:           os : Ubuntu                       :
: private_only : False                        :
:  private_cpu : False                        :
:      created : 2013-06-13T08:29:44-06:00    :
:     modified : 2013-06-13T08:31:57-06:00    :
```

```
:        users : root ABCDEFGH             :
:..............:..........................:
```

There are many other commands to help manage virtual servers. To see them all, use *slcli help vs*.

```
$ slcli vs
Usage: slcli vs [OPTIONS] COMMAND [ARGS]...

  Virtual Servers.

Options:
  --help  Show this message and exit.

Commands:
  cancel          Cancel virtual servers.
  capture         Capture SoftLayer image.
  create          Order/create virtual servers.
  create-options  Virtual server order options.
  credentials     List virtual server credentials.
  detail          Get details for a virtual server.
  dns-sync        Sync DNS records.
  edit            Edit a virtual server's details.
  list            List virtual servers.
  network         Manage network settings.
  pause           Pauses an active virtual server.
  power_off       Power off an active virtual server.
  power_on        Power on a virtual server.
  ready           Check if a virtual server is ready.
  reboot          Reboot an active virtual server.
  reload          Reload operating system on a virtual server.
  rescue          Reboot into a rescue image.
  resume          Resumes a paused virtual server.
  upgrade         Upgrade a virtual server.
```

Configuration Setup

To update the configuration, you can use *slcli setup*.

```
$ slcli setup
Username []: username
API Key or Password []:
Endpoint (public|private|custom): public
:..............:..................................................................:
:         Name : Value                                                            :
:..............:..................................................................:
:     Username : username                                                         :
:      API Key : oyVmeipYQCNrjVS4rF9bHWV7D75S6pa1fghFl384v7mwRCbHTfuJ8qRORIqoVnha :
: Endpoint URL : https://api.softlayer.com/xmlrpc/v3/                            :
:..............:..................................................................:
Are you sure you want to write settings to "/home/me/.softlayer"? [y/N]: y
```

To check the configuration, you can use *slcli config show*.

```
$ slcli config show
:..............:..................................................................:
```

```
:         Name : Value                                                          :
:..............:......................................................................:
:     Username : username                                                       :
:      API Key : oyVmeipYQCNrjVS4rF9bHWV7D75S6pa1fghFl384v7mwRCbHTfuJ8qRORIqoVnha :
: Endpoint URL : https://api.softlayer.com/xmlrpc/v3/                           :
:..............:......................................................................:
```

To see more about the config file format, see *Configuration File*.

Usage Examples

To discover the available commands, simply type *slcli*.

```
$ slcli
Usage: slcli [OPTIONS] COMMAND [ARGS]...

  SoftLayer Command-line Client

Options:
  --format [table|raw|json|jsonraw]
                                  Output format  [default: table]
  -C, --config PATH               Config file location  [default:
                                  ~/.softlayer]
  -v, --verbose                   Sets the debug noise level, specify multiple
                                  times for more verbosity.
  --proxy TEXT                    HTTP[S] proxy to be use to make API calls
  -y, --really / --not-really     Confirm all prompt actions
  --demo / --no-demo              Use demo data instead of actually making API
                                  calls
  --version                       Show the version and exit.
  -h, --help                      Show this message and exit.

Commands:
  block           Block Storage.
  call-api        Call arbitrary API endpoints.
  cdn             Content Delivery Network.
  config          CLI configuration.
  dns             Domain Name System.
  file            File Storage.
  firewall        Firewalls.
  globalip        Global IP addresses.
  hardware        Hardware servers.
  image           Compute images.
  loadbal         Load balancers.
  messaging       Message queue service.
  metadata        Find details about this machine.
  nas             Network Attached Storage.
  object-storage  Object Storage.
  report          Reports.
  rwhois          Referral Whois.
  setup           Edit configuration.
  shell           Enters a shell for slcli.
  sshkey          SSH Keys.
  ssl             SSL Certificates.
  subnet          Network subnets.
  summary         Account summary.
```

```
  ticket          Support tickets.
  virtual         Virtual Servers.
  vlan            Network VLANs.

  To use most commands your SoftLayer username and api_key need to be
  configured. The easiest way to do that is to use: 'slcli setup'
```

As you can see, there are a number of commands/sections. To look at the list of subcommands for virtual servers type *slcli vs*. For example:

```
$ slcli vs
Usage: slcli vs [OPTIONS] COMMAND [ARGS]...

  Virtual Servers.

Options:
  --help  Show this message and exit.

Commands:
  cancel          Cancel virtual servers.
  capture         Capture SoftLayer image.
  create          Order/create virtual servers.
  create-options  Virtual server order options.
  credentials     List virtual server credentials.
  detail          Get details for a virtual server.
  dns-sync        Sync DNS records.
  edit            Edit a virtual server's details.
  list            List virtual servers.
  network         Manage network settings.
  pause           Pauses an active virtual server.
  power_off       Power off an active virtual server.
  power_on        Power on a virtual server.
  ready           Check if a virtual server is ready.
  reboot          Reboot an active virtual server.
  reload          Reload operating system on a virtual server.
  rescue          Reboot into a rescue image.
  resume          Resumes a paused virtual server.
  upgrade         Upgrade a virtual server.
```

Finally, we can make an actual call. Let's list out the virtual servers on our account by using *slcli vs list*.

```
$ slcli vs list
:.........:...........:..................:......:........:................:.......
↪.......:..................:
:    id   : datacenter :       host         : cores : memory :   primary_ip   : ␣
↪backend_ip  : active_transaction :
:.........:...........:..................:......:........:................:.......
↪.......:..................:
: 1234567 :   sjc01    :  test.example.com  :   4   :   4G   :    12.34.56    :   65.
↪43.21   :         -          :
:.........:...........:..................:......:........:................:.......
↪.......:..................:
```

Most commands will take in additional options/arguments. To see all available actions, use *–help*.

```
$ slcli vs list --help
Usage: slcli vs list [OPTIONS]
```

```
  List virtual servers.

Options:
  --sortby [guid|hostname|primary_ip|backend_ip|datacenter]
                                  Column to sort by
  -c, --cpu INTEGER               Number of CPU cores
  -D, --domain TEXT               Domain portion of the FQDN
  -d, --datacenter TEXT           Datacenter shortname
  -H, --hostname TEXT             Host portion of the FQDN
  -m, --memory INTEGER            Memory in mebibytes
  -n, --network TEXT              Network port speed in Mbps
  --hourly                        Show only hourly instances
  --monthly                       Show only monthly instances
  --tags TEXT                     Show instances that have one of these comma-
                                  separated tags
  --help                          Show this message and exit.
```

CHAPTER 5

Contributing

Contribution Guide

This page explains how to get started contributing code to the SoftLayer API Python Bindings project.

Code Organization

- **docs** - Where The source to this documentation lives.
- **SoftLayer** - All the source lives under here.
  - **API** - Primary API client.
  - **CLI** - Code for the command-line interface.
  - **managers** - API Managers. Abstractions to help use the API.

Setting Up A Dev Environment

Before working with the SoftLayer Python API client source, we strongly recommend that you know how to use Python's virtual environment, virtualenv. Virtualenv allows you to create isolated Python environments that are individually tailored to particular development projects. Each environment can have its own set of libraries and even its own Python interpreter. This keeps them fully isolated, reducing the possibility of library conflicts between different projects.

After you have virtualenv, you should set up a virtual environment and activate it whenever you are working on softlayer-python. The commands needed to setup an environment and activate it might look something like this:

```
virtualenv --no-site-packages softlayer_env
source softlayer_env/bin/activate
```

Please refer to the virtualenv documentation for more information about creating, and working with virtual environments.

Once you have an appropriate environment, you will then download the SoftLayer API Python Bindings source code by following the *installation instructions*. Change into softlayer-python source directory and run the following to install the pre-requisites that you'll need in order to run the test suites:

```
pip install -r tools/test-requirements.txt
```

Testing

The project has a mix of functional and unit tests. Before submitting changes to be integrated into the project, you should validate your code using tox. Simply issue the tox command from the root of the source tree:

```
tox
```

In addition to testing different versions of Python, tox checks for common mistakes in the code using Flake8 and pylint. You should eliminate the linting errors that are reported before submitting your code. You can run only the linting checks by using this command:

```
tox -eanalysis
```

The project's configuration instructs tox to test against many different versions of Python. A tox test will use as many of those as it can find on your local computer. Rather than installing all those versions, we recommend that you point the Travis continuous integration tool at your GitHub fork. Travis will run the test against the full suite of Python versions every time you push new code.

Using tox to run tests in multiple environments can be very time consuming. If you wish to quickly run the tests in your own environment, you may do so using py.test. The command to do that is:

```
py.test tests
```

Documentation

The project is documented in reStructuredText and built using Sphinx. If you have fabric installed, you simply need to run the following to build the docs:

```
fab make_html
```

The documentation will be built in *docs/_build/html*. If you don't have fabric, use the following commands.

```
cd docs
make html
```

The primary docs are built at Read the Docs.

Style

This project tries to follow **PEP 8** and most of the style suggestions that pyflakes recommends. Run Flake8 regularly. Flake8, with project-specific exceptions, can be run by using tox:

```
tox -e analysis
```

Contributing

Contributing to the Python API bindings follows the fork-pull-request model on GitHub. The project uses GitHub's issue tracker and pull requests to manage source control, bug fixes and new feature development regarding the API bindings and the CLI. In order to contribute, we require that you sign a contributer agreemenet:

- Sign our contributor agreement (CLA) You can find the `CLA here`.
- If you're contributing on behalf of your employer we'll need a signed copy of our corporate contributor agreement (CCLA) as well. You can find the `CCLA here`.

Developer Resources

Command-Line Interface Developer Guide

The SoftLayer CLI can be used to manage many different SoftLayer services directly from the command line.

The command line parsing is currently based on click, which is a command parsing library along with some additions to dynamically load modules from a routes-like file and from entry points.

First Example

For the first example, we can create *slcli table-example* by creating the following file at SoftLayer/CLI/table_example.py:

```
"""A formatting table example."""
from SoftLayer.CLI import environment
from SoftLayer.CLI import formatting

import click


@click.command()
@environment.pass_env
def cli(env):
    """This returns an table that highlights how tables are output"""
    # create a table with two columns: col1, col2
    table = formatting.Table(['col1', 'col2'])

    # align the data facing each other
    # valid values are r, c, l for right, center, left
    # note, these are suggestions based on the format chosen by the user
    table.align['col1'] = 'r'
    table.align['col2'] = 'l'

    # add rows
    table.add_row(['test', 'test'])
    table.add_row(['test2', 'test2'])

    env.fout(table)
```

Then we need to register it so that *slcli table-example* will know to route to this new module. We do that by adding ALL_ROUTES in SoftLayer/CLI/routes.py to include the following:

```
...
('table-example', 'SoftLayer.CLI.table_example:cli'),
...
```

Which gives us

```
$ slcli table-example
:.......:.......:
:  col1 : col2  :
:.......:.......:
:  test : test  :
: test2 : test2 :
:.......:.......:

$ slcli --format=raw table-example
 test   test
 test2  test2
```

Formatting of the data represented in the table is actually controlled upstream from the CLIRunnable's making supporting more data formats in the future easier.

Arguments

A command usually isn't very useful without context or arguments of some kind. With click, you have a large array of argument and option types at your disposal. Additionally, with the SoftLayer CLI, we have global options and context which is stored in *SoftLayer.CLI.environment.Environment* and is attainable through a decorator located at *SoftLayer.CLI.environment.pass_env*. An example of options and the environment is shown below. It also shows how output should be done using *env.out* instead of printing. This is used for testing and to have a consistent way to print things onto the screen.

```
from SoftLayer.CLI import environment

import click


@click.command()
@click.option("--number",
              required=True,
              type=click.INT,
              help="print different output")
@click.option("--choice",
              type=click.Choice(['this', 'that']),
              help="print different output")
@click.option("--test", help="print different output")
@environment.pass_env
def cli(env, number, choice, test):
    """Argument parsing example"""

    if test:
        env.out("Just testing, move along...")
    else:
        env.out("This is fo'realz!")

    if choice == 'this':
        env.out("Selected this")
    elif choice == 'that':
```

```
        env.out("Selected that")

    env.out("This is a number: %d" % number)
```

Refer to the click library documentation for more options.

Accessing the API

A SoftLayer client is stood up for every command and is available through *SoftLayer.CLI.environment.Environment.client*. The example below shows how to make a simple API call to the SoftLayer_Account::getObject.

```
from SoftLayer.CLI import environment

import click


@click.command()
@environment.pass_env
def cli(env):
    """Using the SoftLayer API client"""

    account = env.client['Account'].getObject()
    return account['companyName']
```

Aborting execution

When a confirmation fails, you probably want to stop execution and give a non-zero exit code. To do that, raise a *SoftLayer.CLI.exceptions.CLIAbort* exception with the message for the user as the first parameter. This will prevent any further execution and properly return the right error code.

```
raise CLIAbort("Aborting. Failed confirmation")
```

CHAPTER 6

External Links

- SoftLayer API Documentation
- Source on GitHub
- Issues
- Pull Requests
- PyPI
- Twitter
- #softlayer on freenode

Python Module Index

S

Index

D

E

F

G

H

I

L

M

N

P

Q

R

S

T

U

V

W