bss_api Documentation

Release 0.7

Cathal Dinneen

Mar 10, 2019

Contents

1 Getting Started **1**

2 API Documentation **3**
- 2.1 Organizations . . . 3
- 2.2 Subscriber . . . 8
- 2.3 Subscriptions . . . 11
- 2.4 Seats . . . 13
- 2.5 Contact . . . 14
- 2.6 Address Set . . . 15
- 2.7 Enums . . . 15
- 2.8 Exceptions . . . 18

3 Examples **19**
- 3.1 Activate any user in a Pending state and set a temporary password . . . 19
- 3.2 Returns subscribers that were modified within the last 7 days . . . 19
- 3.3 Create an organisation, add Connections S2 and ICEC subscriptions . . . 20
- 3.4 Get print each user in an org in the format username, city . . . 20

4 F.A.Q. **21**
- 4.1 What IBM Smart Cloud BSS? . . . 21
- 4.2 What is needed to use this code? . . . 21
- 4.3 Will this work on an account created through a third party/IBM Martketplace . . . 21

Python Module Index **23**

CHAPTER 1

Getting Started

Install through pip

```
pip install smartcloudadmin
```

Retrieving an Organization

```
from smartcloudadmin import Organization
from smartcloudadmin.bss_config import BssConfig

# Provide North American data center details and supply user credentials
config = BssConfig()
config.add_datacenter("NA", "https://apps.na.collabserv.com", (email_address,␣
↪password))

# Create an Organization object for organization id 11111111 on NA
my_org = Organization.get("NA", "11111111")
```

Entitle a user with a subscription

```
from smartcloudadmin import Subscriber
# Get user from id
new_subscriber = Subscriber.get("NA", 222222)
# Or alternatively through email address
# new_subscriber = Subscriber.get("NA", email="test_email@ibm.com")

# Entitle the user with subscription Id 33333333
new_subscriber.entitle(33333333)
```

CHAPTER 2

API Documentation

2.1 Organizations

class smartcloudadmin.models.organization.**Organization**(*environment: str, id: int = 0, name: str = '', address_set: smartcloudadmin.models.address_set.AddressSet = '', contact: smartcloudadmin.models.contact.Contact = '', language_preference: str = 'en_US', state: str = '', time_zone: str = 'America/Central', payment_method_type: str = 'PURCHASE_ORDER', currency_type: str = 'USD', owner: int = 0, created: datetime.datetime = '01/01/1970 00:00:00', modified: datetime.datetime = '01/01/1970 00:00:00', party_type: str = <PartyType.ORGANISATION: 'ORGANIZATION'>, security_realm: str = 'NON_FEDERATED', industry=''*)

Bases: object

Represents an Organization within IBM Social CLoud.

Supported Operations:

Operation	Description
x == y	Test whether 2 Organization objects contain the same data
x != y	Test whether 2 Organization objects do not contain the same data
str(x)	Returns basic details about the Organization

id [int] The id of the Organization. May be referred to as Customer Id/

name [str] Name of the Organization

environment [str] location where the organization is hosted. **I.E.** North America, Central Europe or Asia Pacific

contact [*Contact*] Object containing details about Organization owner.

address_set [*AddressSet*] Object containing physical address of the Organization.

language_preference [*LanguagePreference*] Language preference when interacting with BSS through UI.

state [*State*] Current state of the Organization. **I.E.** Active

time_zone [*TimeZone*] Timezone the Organization is located in.

payment_method_type [*PaymentMethodType*] Payment method used by Organization

currency_type [*CurrencyType*] Currency the Organization pays in.

created [Datetime] Creation time for the organization given in the format of %Y/%m/%d %H:%M:%S

modified [Datetime] Modification time for the organization given in the format of %Y/%m/%d %H:%M:%S

party_type [*PartyType*] Party Type

security_realm [*SecurityRealm*] Security configuration of the Organization. I.E. Federated/Non-federated.

subscriptions [{Subscription}] A dictionary of Subscriber objects where the key is the Subscriber Id

subscribers [{Subscriber}] A dictionary of Subscription objects where the key is the Subscription Id

admins: {Subscriber} A dictionary of Subscription objects where the key is the Subscription Id

size [str] A range of the number of users in an Organization. i.E 100<500

industry [str] The business or industry the Organization is in.

vendor_id [int] Vendor id of the Organization. I.E. 10.

is_guest [bool] Current state of the Organization. **I.E.** Active

customer_type [*CustomerType*] Current state of the Organization. **I.E.** Active

is_partner [bool] Whether or not the Organization is an IBM partner.

is_sync_pending [bool] Current state of the Organization. **I.E.** Active

last_sync_date [bool] Current state of the Organization. **I.E.** Active

add_subscriber(*, *email_address*, *given_name*, *family_name*, ***kawrgs*) → smartcloudadmin.models.subscriber.Subscriber

Adds a new user to the Organization.

For a full list of optional params see `bssapi.models.subscriber.Subscriber.create()`

Parameters

- **email_address** – User's email address

- **given_name** – User's first name
- **family_name** – User's surname

Raises PermissionError: User is not authorised to execute this request.

Returns Newly created Subscriber.

Return type *Subscriber*

Example

```
>>> Subscriber.create(given_name="Tim", family_name="Tom" ,email_address="tim.
↪tom@tam.net")
```

add_subscription(*, *part_number*, *part_quantity*, *duration_length*, *duration_units*) → smartcloudadmin.models.subscription.Subscription

Adds a new subscription to an Organization.

For a full list of optional params see `bssapi.models.subscriber.Subscription.create()`

Parameters

- **part_number** – Subscription part number. I.E. D0NPULL for Connections Cloud.
- **part_quantity** – Number of seats
- **duration_length** – length of Subscription duration units
- **duration_units** – options YEARS or MONTHS (possibly DAYS)

Returns Created Subscription object

Return type *Subscription*

Raises **PermissionError** – User is not authorised to execute this request.

:example :

```
>>> my_organisation.add_subscription(part_number="D0NPULL", part_
↪quantity=25, duration_length=1,
duration_units="YEARS")
```

check_for_updates() → bool

Compares current Organization object with live server data and updates if there are differences.

Returns **if** an update was made

Return type bool

Example

```
>>> my_organization.check_for_updates()
```

country

Country of which the Organization resides as per address_set.

Returns Country the Organization is located in

Return type str

classmethod create(*environment: str*, *organisation_name: str*, *given_name: str*, *family_name: str*, *admin_email: str*, ***kwargs*) → smartcloudadmin.models.organization.Organization

Creates a new organisation on BSS and returns organisation object.

Parameters

- **environment** – Environment of the organisation , e.g. A3,G3,S3
- **organisation_name** – Name of the organisation.
- **given_name** – First name of the organisation owner.
- **family_name** – Surname of the organisation owner.
- **admin_email** – Email address for the organisation admin

Returns an organisation object

Raises PermissionError: User is not authorised to execute this request.

Return type *Organization*

Example

```
>>> resp = bss_api.create_org(environment, body)
```

delete() → None

Deletes the Organization.

Raises PermissionError: User is not authorised to execute this request.

Example

```
>>> my_organization.delete()
```

filter_subscribers(*, *attribute*, *attribute_value*, *passed_operator=<built-in function eq>*) → {<class 'smartcloudadmin.models.subscriber.Subscriber'>}

Returns a filtered subscriber list based instance attributes and queried attribute value. For greater than / less than queries it makes sense to think of it as : Is subscriber.modified date greater (newer) than attribute_value?

Parameters

- **attribute** – Subscriber attribute that is being filtered.
- **attribute_value** – Value of attribute you want to filter against.
- **passed_operator** – Operator to be carried out agaisnt attribute and value. E.G. State is not equal to ACTIVE.

Returns [Subscriber]

Raises **PermissionError** – User is not authorised to execute this request.

:example : my_organisation.filter_subscribers(attribute="state", attribute_value="ACTIVE", passed_operator=operator.ne)

filter_subscriptions(*, *attribute*, *attribute_value*, *passed_operator=<built-in function eq>*) → {<class 'smartcloudadmin.models.subscription.Subscription'>}

Returns a filtered subscriber list based instance attributes and queried attribute value. For greater than / less than queries it makes sense to think of it as : Is subscription.modified date greater (newer) than attribute_value?

Parameters

- **attribute** – Subscription attribute that is being filtered.
- **attribute_value** – Value of attribute you want to filter against.

- **passed_operator** – Operator to be carried out agaisnt attribute and value. E.G. State is

not equal to ACTIVE. :returns: A List of subscriptions that match the given query :rtype: {Subscription} :raises PermissionError: User is not authorised to execute this request.

Example

```
>>> my_organisation.filter_subscriptions(attribute="state",␣
↪attribute_value="ACTIVE",passed_operator=operator.ne)
```

classmethod from_json(*environment*, *json_body*) → smartcloudadmin.models.organization.Organization

Creates an Organization object from a JSON payload, for example when retrieving many orgs or organization searches :param environment: Datacenter :param json_body: JSON Payload :return: Organization :rtype: Organization

classmethod get(*environment: str*, *organization_id: int*) → smartcloudadmin.models.organization.Organization

Creates a new organisation on BSS and returns that organisation object.

Parameters

- **environment** – Environment of the organisation , e.g. NA, CE, AP
- **organization_id** – Name of the organisation.

Returns Retrieved Organization

Return type *Organization*

Raises PermissionError: User is not authorised to execute this request.

Example >>>resp = bss_api.create_org(environment, body)

classmethod get_basic(*environment: str*, *organization_id: int*) → smartcloudadmin.models.organization.Organization

Gets an Organization object with basic Organization details. Does not retrieve subscription or subscriber information. :param environment: Datacenter where the Organization resides :param organization_id: Organization id :return: a Basic Organization :rtype: Organization

remove_subscriber(*subscriber: smartcloudadmin.models.subscriber.Subscriber*) → None

Removes the Subscriber

Parameters **subscriber** –

Returns None

Example

```
>>> tom = Subscriber.get("A3", 2142424)
>>> my_organization.remove_subscriber(tom)
```

remove_subscription(*subscription: smartcloudadmin.models.subscription.Subscription*) → None

Returns a filtered subscriber list based instance attributes and queried attribute value. For greater than / less than queries it makes sense to think of it as : Is subscriber.modified date greater (newer) than attribute_value?

Parameters **subscription** – Subscription to delete

Raises **PermissionError** – User is not authorised to execute this request.

```
>>> connections_subscription = Subscription.get("A3",121242)
>>> my_organization.remove_subscription(connections_subscription)
```

subscriber_count

subscription_count
The number of subscriptions the Organization has.

Returns The number of Subscriptions

Return type int

suspend() → None
Suspends the Organization.

Raises PermissionError: User is not authorised to execute this request.

Example

```
>>> my_organization.suspend()
```

unsuspend() → None
Unsuspends the Organization.

Raises PermissionError: User is not authorised to execute this request.

Example

```
>>> my_organization.unsuspend()
```

2.2 Subscriber

class smartcloudadmin.models.subscriber.**Subscriber**(*environment: str*, ***, *customer_id: int = 0*, *org_name: str = ''*, *owner: str = 0*, *modified: datetime.datetime = '01/01/1970 00:00:00'*, *is_guest: bool = False*, *created: datetime.datetime = '01/01/1970 00:00:00'*, *subscriber_state: str = ''*, *party_role_type=None*, *deleted: datetime.datetime = '01/01/1970 00:00:00'*, *role_set=None*, *id: int = 0*, *email: str = ''*, *given_name: str = ''*, *family_name: str = ''*, *name_prefix: str = ''*, *name_suffix: str = ''*, *security_realm: smartcloudadmin.enums.State = ''*, *employee_number: str = ''*)

Bases: object

Represents an Subscriber used within IBM Social Cloud. Contains details about the physical location of the Organization headquarters.

environment [str] Environment/Datacenter the Organization resides on.

email [str] Subscriber's email address.

given_name [str] First name of subscriber.

family_name [str] Surname of subscriber.

id [int] Subscriber id

org_name [str] Name of Organization to which subscriber belongs to.

owner [int] Owner id of Organization.

created [Datetime] Time of subscriber creation.

modified [enumerate] Time of subscriber was last modified.

is_guest [bool] Whether the subscriber is a guest of the Organization or not.

state [str] Current user activity state.

party_role_type [str] State where Organization resides.

deleted [bool] Is the subscriber deleted.

customer_id [str] Organization id

role_set [str] List of permissions the user has. E.G. User, Application Developer, Administrator.

name_prefix [str] Honorary Subscriber prefix. I.E. Dr.

name_suffix [str] Name Suffic, I.E. Jr.

security_realm [enumerate] Subscriber's federation type.

employee_number [str] Organization's number for subscriber

seat_set [[Seat]] List of Seat objects which subscriber occupies.

entitlements [[int]] List of subscription id's to which subscriber has a seat.

activate() → None
Activates the user on smartcloud. :return: None

assign_role(*valid_role*) → None
Assigns a role to a subscriber. :param valid_role:

change_password(*current_password*, *password*) → None
Changes the subscriber's password. Need to know current password in order to change. Could be used in conjunction with set_one_time_password to initially set and then change a password. :param current_password: Subscriber's current password :param password: New password for the subscriber.

classmethod create(*environment*, *organization_id*, *org_name*, ***, *email_address*, *given_name*, *family_name*, ***kwargs*) → smartcloudadmin.models.subscriber.Subscriber
Creates a subscriber object.

See also Organzation.add_subscriber() Required ——— :param environment: Datacenter to create the Organization on. :param organization_id: ID of subscribers organiation :param org_name: Name of subscribers organisation. :param email_address: Subscriber's Email address Optional ——— :param given_name: Subscriber's first name, default : "" :param family_name: Subscriber's surname,default : "" :param role_set: Role Subscriber should be granted on creation, defaults to User. :param name_prefix: default : "" :param name_suffix: default : "" :param employee_number: default : "" :param language_preference: Defaults to English :param work_phone: default : "" :param mobile_phone: default : "" :param home_phone: default : "" :param fax: default : "" :param job_title: default : "" :param website_address: default : "" :param time_zone: Defaults to GMT :param photo: default : ""

Returns Subscriber

delete(*, *soft_delete=True*) → None
Delete subscriber. :param soft_delete: Should the subscriber be soft deleted or hard deleted. By default the user is soft deleted. :type bool:

entitle(*subscription_id*) → None
Entitles a subscriber with a subscription from the organization. :param subscription_id: subscription id to entitle user with.

classmethod from_json(*environment*, *json_body*) → smartcloudadmin.models.subscriber.Subscriber

classmethod get(*environment*, *, *subscriber_id=None*, *email_address=None*) → smartcloudadmin.models.subscriber.Subscriber

get_role_list()

get_roles() → [<class 'str'>]
The Roles currently held by subscriber. :return: he current roles groups a subscriber is a member of.

name

Returns Subscriber's full formal name.

reset_password() → None

restore() → None
Restore a user from a soft deleted state.

revoke(*subscription_id*) → None
Revokes a subcribers enttilement to a subscription. :param subscription_id:w

set_one_time_password(*password*) → None
Sets a one time password for the subscriber which they will then be prompted to change on login. :param password:

set_password(*password*) → None
Sets subscribers password. :param password: Password to set.

show_as_row() → str

Returns Basic details relevent about a user.

show_as_summary() → str

suspend() → None
Suspends a subscriber.

unassign_role(*valid_role*) → None
Removes a role from a subscriber. :param valid_role:

unsuspend() → None
Unsuspends a subscriber.

2.3 Subscriptions

class smartcloudadmin.models.subscription.**Subscription**(*environment: str, customer_id: int = 0, part_number='', id=0, part_quantity=0, state='', available_seats=0, modified='01/01/1970 00:00:00', created='01/01/1970 00:00:00', duration_length=0, duration_unit=0, entitlement_quantity_available=0, max_number_of_seats=0, is_automatically_renewed=False, purchase_date='01/01/1970 00:00:00', is_trial=False, deleted=False, is_beta=False, is_free=False, parent_subscription_id=0, billing_frequency='', effective_date='01/01/1970 00:00:00', expiration_date='01/01/1970 00:00:00'*)

Bases: object

Represents an 'Subscription <https://www-10.lotus.com/ldd/appdevwiki.nsf/xpAPIViewer.xsp?lookupName=API+Reference#action=openDocument&res_title=Create_subscription_bss&content=apicontent>'_within IBM Social Cloud. Supported Operations:

Operation	Description
x == y	Test whether 2 Contact objects contain the same data
x != y	Test whether 2 Contact objects do not contain the same data
str(x)	Returns basic details about the Contact

environment [str] Datacenter where the Subscription resides

part_number [str] Subscription part_number, e.g. D0NPULL, D0NRILL

id [int] Subscription id

customer_id [id] Organization id

created [Datetime] Creation date for the Subscription.

modified [Datetime] Modification date for the Subscription.

part_quantity [int] Number of Seats to be created for the Subscription

available_numbers_of_seats [int] Unoccupied seats for Subscription

state [str] Current state of Subscription

deleted [bool] Whether the Subscription user still exists.

duration_length [int] Lifespan of Subscription in duration_units

duration_unit [str] Units of time duration_length is measured in. MONTHS/YEARS

entitlement_quantity_available [int] entitlement quantity

max_number_of_seats [int] Max seats on Subscription

is_automatically_renewed [bool] Weather the subscription will renew or expire at expiration date.

purchase_date [Datetime] Purchase date

is_trial [bool] Is Subscription trial

is_beta [bool] Is Subscription beta

is_free [bool] Is Subscription free

parent_subscription_id [str] -1 for standard Subscriptions. Child Subscriptions will have a parent id

billing_frequency [str] Frequency

effective_date [Datetime] Subscription start date

expiration_date [Datetime] Subscription end date

classmethod create(*environment*, *customer_id*, *part_number*, *part_quantity*, *duration_units*, *duration_length*, ***kwargs*) → smartcloudadmin.models.subscription.Subscription

Creates a new subscription

Parameters

- **environment** – Datacenter
- **customer_id** – Organization_id
- **part_number** – Part Number for Subscription
- **part_quantity** – Number of Seats for Subscription
- **duration_units** – Months/Years
- **duration_length** – Duration length in Units
- **kwargs** –

Returns Newly created Subscription

Return type *Subscription*

created_epoch

Returns created date in epoch format

delete() → None

classmethod from_json(*environment*, *json_body*) → smartcloudadmin.models.subscription.Subscription

classmethod get(*environment*, *subscription_id*) → smartcloudadmin.models.subscription.Subscription

Populates subscription with an existing subcription details using BSS API :param environment: Datacenter Subscription resides on :type: str :param subscription_id: subscription id to retrieve :type: int :returns: Subscription :rtype: Subscription

modified_epoch

Returns modified date in epoch format

show_as_row() → str

show_as_summary() → str

suspend() → None

transfer_seat(*seat_id*, *target_subscription*) → None

unsuspend() → None

2.4 Seats

class smartcloudadmin.models.seat.**Seat**

Bases: object

Represents an Seat within an IBM Social CLoud Subscription.

Supported Operations:

Operation	Description
x == y	Test whether 2 Seat objects contain the same data
x != y	Test whether 2 Seat objects do not contain the same data
str(x)	Returns basic details about the Seat

owner [int] Id of the seat owner.

terms_of_user_id [int] update_this

vendor_id [int] Id of the Organization Vendor.

seat_state [*State*] Users fully quantified domain name as it is in IBM SmartCloud's LDAP server.

created [Datetime] Creation date for the Seat record.

modified [Datetime] Modification date for the Seat record.

subscription_id [int] Subscription id of the seat instance

subscriber_id [int] Subscriber id of the seat instance

entitlement_quantity_allocated [int] update_this

version [int] update_this

provisioning_workflow_id [int] update_this

seat_service_product_attribute_set [str] update_this

workflow_id_list [str] update_this

deleted [bool] Is the seat deleted

id [int] Seat Id

has_accepted_terms_of_use [int] update_this

classmethod from_json(*json_body*) → smartcloudadmin.models.seat.Seat

2.5 Contact

class smartcloudadmin.models.contact.**Contact**(*, *given_name*, *family_name*, *email_address*, *ldap_dn: str = ''*, *employee_number: str = ''*, *created: datetime.datetime = '01/01/1970 00:00:00'*, *modified: datetime.datetime = '01/01/1970 00:00:00'*, *name_prefix: str = ''*, *name_suffix: str = ''*, *deleted: bool*, *org_name: str = ''*, *org_id: int = 0*, *security_realm: smartcloudadmin.enums.SecurityRealm = <SecurityRealm.NON_FEDERATED: 'NON_FEDERATED'>*, *time_zone: str = 'America/Central'*, *job_title: str = 'Employee'*, *mobile_phone: str = '0000000'*, *work_phone: str = '0000000'*, *home_phone: str = '0000000'*)

Bases: object

Represents an Organization within IBM Social CLoud.

Supported Operations:

Operation	Description
str(x)	Returns basic details about the Contact

given_name [str] First name of the contact.

family_name [str] Surname of the contact.

email_address [str] Login email address of the user.

ldap_dn [str] Users fully quantified domain name as it is in IBM SmartCloud's LDAP server.

created [Datetime] Creation date for the Contact record.

modified [Datetime] Modification date for the Contact record.

employee_number [str] Organization supplied employee number.

name_prefix [str] Honorary title I.E. Mr. Ms. Dr.

name_suffix [str] Name Suffix I.E. Jr.

deleted [bool] Whether the Contact user still exists.

security_realm [*SecurityRealm*] Security configuration for the Contact.

time_zone [*TimeZone*] Timezone of the Contact.

job_title [str] Job title of the Contact.

mobile_phone [str] Mobile phone number to reach Contact.

work_phone [str] Work phone number to reach Contact.

home_phone [str] Home phone number to reach Contact.

classmethod from_json()

2.6 Address Set

class smartcloudadmin.models.address_set.**AddressSet**(*, *state_code*, *postal_code*, *city*, *state*, *country*, *country_code*, *address_type*, *modified*, *address_line_1=''*, *address_line_2=''*)

Bases: object

Represents an Address Set used within IBM Social CLoud. Contains details about the physical location of the Organization headquarters.

state_code [enumerate] State where Organization resides.

postal_code [str] Postal code of Organization

city [str] City where Organization resides.

modified [Datetime] Modification date of the Address Set record

address_line_1 [str] Language preference when interacting with BSS through UI.

address_line_2 [str] State where Organization resides.

state [str] State where Organization resides.

country [enumerate] Country where Organization resides.

country_code [str] Country Code where Organization resides.

address_type [enumerate] Type of Address for record.

classmethod from_json(*, *modified*)

Creates a dummy address set object for situations where no address set exists. Modified value passed through to see if an address set is added at a later date.

Parameters **modified** –

Returns address_set

classmethod not_provided(*, *modified*)

2.7 Enums

class smartcloudadmin.enums.**AddressType**

Bases: enum.Enum

Used to differentiate Billing and Mailing address sets

BILLING = 'BILLING'

MAILING = 'MAILING'

MULTIPURPOSE = 'MULTIPURPOSE'

class smartcloudadmin.enums.**BSSBoolean**

Bases: enum.Enum

An enumeration.

FALSE = 'false'

TRUE = 'true'

class smartcloudadmin.enums.**CurrencyType**
Bases: enum.Enum

Supported currencies for payment.

AUD = 'AUD'

CAD = 'CAD'

EUR = 'EUR'

GBP = 'GBP'

INR = 'INR'

JPY = 'JPY'

NZD = 'NZD'

USD = 'USD'

WON = 'WON'

class smartcloudadmin.enums.**CustomerIdType**
Bases: enum.Enum

CAAS_CUSTOMER_ID = 'CAAS_CUSTOMER_ID'

COREMETRIC_CUSTOMER_ID = 'COREMETRIC_CUSTOMER_ID'

GLOBALCROSSING_ID = 'GLOBALCROSSING_ID'

GLOBALIVE_ID = 'GLOBALIVE_ID'

IBM_CUSTOMER_NUMBER = 'IBM_CUSTOMER_NUMBER'

IBM_CUSTOMER_NUMBER_PREV = 'IBM_CUSTOMER_NUMBER_PREV'

IBM_SITE_NUMBER = 'IBM_SITE_NUMBER'

SALESFORCE_ACCOUNT_ID = 'SALESFORCE_ACCOUNT_ID'

SALESFORCE_CONTACT_ID = 'SALESFORCE_CONTACT_ID'

SALESFORCE_ID = 'SALESFORCE_ID'

SALESFORCE_LEAD_ID = 'SALESFORCE_LEAD_ID'

SALESFORCE_OPPORTUNITY_ID = 'SALESFORCE_OPPORTUNITY_ID'

SIEBEL_ID = 'SIEBEL_ID'

STERLING_CUSTOMER_ID = 'STERLING_CUSTOMER_ID'

TMS_CUSTOMER_ID = 'TMS_CUSTOMER_ID'

UNICA_CUSTOMER_ID = 'UNICA_CUSTOMER_ID'

UNYTE_CUSTOMER_ID = 'UNYTE_CUSTOMER_ID'

class smartcloudadmin.enums.**CustomerType**
Bases: enum.Enum

An enumeration.

DIRECT = 'DIRECT'

class smartcloudadmin.enums.**LanguagePreference**
Bases: enum.Enum

Security configuration used by the Organization/Subscriber

EN_US = 'en_US'

class smartcloudadmin.enums.**PartyRollType**
Bases: enum.Enum

An enumeration.

SUBSCRIBER = 'SUBSCRIBER'

class smartcloudadmin.enums.**PartyType**
Bases: enum.Enum

An enumeration.

ORGANISATION = 'ORGANIZATION'

PERSON = 'PERSON'

class smartcloudadmin.enums.**PaymentMethodType**
Bases: enum.Enum

Supported payment options for IBM Smart Cloud

CREDIT_CARD = 'CREDIT_CARD'

INVOICE = 'INVOICE'

NONE = 'NONE'

PURCHASE_ORDER = 'PURCHASE_ORDER'

class smartcloudadmin.enums.**RoleSet**
Bases: enum.Enum

Supported roles that can be assigned.

APP_DEVELOPER = 'AppDeveloper'

CONNECTIONS_AUDITOR = 'ConnectionsAuditor'

CONTENT_VALIDATOR = 'ContentValidator'

CUSTOMER_ADMINISTRATOR = 'CustomerAdministrator'

DATA_MIGRATOR = 'DataMigrator'

SUPPORT = 'Support'

USER = 'User'

USER_ACCOUNT_ASSISTANT = 'UserAccountAssistant'

VSR = 'VSR'

class smartcloudadmin.enums.**SecurityRealm**
Bases: enum.Enum

Security configuration used by the Organization/Subscriber

FEDERATED = 'FEDERATED'

MODIFIED_FEDERATED = 'MODIFIED_FEDERATED'

NON_FEDERATED = 'NON_FEDERATED'

PARTIAL_FEDERATED = 'PARTIAL_FEDERATED'

class smartcloudadmin.enums.**State**
Bases: enum.Enum

Supported states for various Organizations,Subscriptions,Subscribers etc.

ACTIVE = 'ACTIVE'

CANCEL_PENDING = 'CANCEL_PENDING'

DEREGISTER_PENDING = 'DEREGISTER_PENDING'

PENDING = 'PENDING'

REMOVE_PENDING = 'REMOVE_PENDING'

SOFT_DELETED = 'SOFT_DELETED'

SUSPENDED = 'SUSPENDED'

UNSET = ''

class smartcloudadmin.enums.**TimeZone**
Bases: enum.Enum

Available timezones for use in IBM SmartCloud

ALASKA_STANDARD_TIME = 'AST'

ATLANTIC_STANDARD_TIME = 'AST'

CENTRAL_STANDARD_TIME = 'CST'

EASTERN_STANDARD_TIME = 'EST'

HAWAII_STANDARD_TIME = 'HST'

MOUNTAIN_STANDARD_TIME = 'MST'

NEWFOUNDLAND_STANDARD_TIME = 'NST'

PACIFIC_STANDARD_TIME = 'PST'

YUKON_STANDARD_TIME = 'YST'

2.8 Exceptions

CHAPTER 3

Examples

3.1 Activate any user in a Pending state and set a temporary password

```
my_org = Organization.get(1213232)
pending_subscribers = my_org.filter_subscribers(attribute="state", attribute_value=
↪"PENDING",passed_operator=operator.eq)
for subscriber in pending_subscribers:
    subscriber.activate()
    subscriber.setOneTimePassword("users_temp_password")
```

3.2 Returns subscribers that were modified within the last 7 days

```
my_org.filter_subscribers(attribute="modified", attribute_value=LAST_WEEK,passed_
↪operator=operator.ge)

for user in my_org.subscribers:
    if user.state != "PENDING":
        print(user.email + " : " + user.state)

this could also be done with a filter
my_org.filter_subscribers(attribute="state", attribute_value="PENDING",passed_
↪operator=operator.ge)
```

3.3 Create an organisation, add Connections S2 and ICEC subscriptions

```
org = Organization.create("A3", "Neat_new_company", "admin_email@ibm.com", "John",
↪"Smith")

# Create a Connections subscription
connections_subscription = org.add_subscription("D0NPULL", "100")

for lp in range(100):
    try:
         subscriber = org.add_subscriber()
         subscriber.entitle(connections_subscription.id)
         subscriber.activate()
         subscriber.set_one_time_password("Test1Test")
         subscriber.change_password("temp_password123", "final_password123")
     except Exception as e:
         logger.warn(e)

admin = Subscriber.get("A3", email_address="admin_email@ibm.com")
admin.activate()
admin.set_one_time_password("temp_password123")
admin.change_password("temp_password123", "final_password123")
```

3.4 Get print each user in an org in the format username, city

```
for subscriber_id,subscriber in org.subscribers.items():
    f = open("user_list .txt", "a")
    f.write(f"{subscriber.email},Pa88w0rd\n")
```

CHAPTER 4

F.A.Q.

4.1 What IBM Smart Cloud BSS?

The BSS handles authentication, authorization and provisioning for IBM SmartCloud components This library is intended for use to aid in the above tasks and not for directly interacting with Connections, Sametime or Notes.

4.2 What is needed to use this code?

An active IBM SmartCloud subscription and an account. To really make the most of this code at least CustomerAdministrator privilege is required in order to manage your Organization.

4.3 Will this work on an account created through a third party/IBM Martketplace

It should work the same for a reseller organization but for an organization created through IBM Marketplace it should be fine to read data but and updates should be done through a separate API which is currently not handled by this code.

Note:

Accounts created through traditional IBM processes will work.

Python Module Index

S

smartcloudadmin.enums, 15
smartcloudadmin.models.address_set, 15
smartcloudadmin.models.contact, 14
smartcloudadmin.models.organization, 3
smartcloudadmin.models.seat, 13
smartcloudadmin.models.subscriber, 8
smartcloudadmin.models.subscription, 11

Index

A

activate() (smartcloudadmin.models.subscriber.Subscriber method), 9
ACTIVE (smartcloudadmin.enums.State attribute), 18
add_subscriber() (smartcloudadmin.models.organization.Organization method), 4
add_subscription() (smartcloudadmin.models.organization.Organization method), 5
AddressSet (class in smartcloudadmin.models.address_set), 15
AddressType (class in smartcloudadmin.enums), 15
ALASKA_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
APP_DEVELOPER (smartcloudadmin.enums.RoleSet attribute), 17
assign_role() (smartcloudadmin.models.subscriber.Subscriber method), 9
ATLANTIC_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
AUD (smartcloudadmin.enums.CurrencyType attribute), 16

B

BILLING (smartcloudadmin.enums.AddressType attribute), 15
BSSBoolean (class in smartcloudadmin.enums), 15

C

CAAS_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
CAD (smartcloudadmin.enums.CurrencyType attribute), 16
CANCEL_PENDING (smartcloudadmin.enums.State attribute), 18
CENTRAL_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
change_password() (smartcloudadmin.models.subscriber.Subscriber method), 9
check_for_updates() (smartcloudadmin.models.organization.Organization method), 5
CONNECTIONS_AUDITOR (smartcloudadmin.enums.RoleSet attribute), 17
Contact (class in smartcloudadmin.models.contact), 14
CONTENT_VALIDATOR (smartcloudadmin.enums.RoleSet attribute), 17
COREMETRIC_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
country (smartcloudadmin.models.organization.Organization attribute), 5
create() (smartcloudadmin.models.organization.Organization class method), 5
create() (smartcloudadmin.models.subscriber.Subscriber class method), 9
create() (smartcloudadmin.models.subscription.Subscription class method), 12
created_epoch (smartcloudadmin.models.subscription.Subscription attribute), 12
CREDIT_CARD (smartcloudadmin.enums.PaymentMethodType attribute), 17
CurrencyType (class in smartcloudadmin.enums), 15
CUSTOMER_ADMINISTRATOR (smartcloudadmin.enums.RoleSet attribute), 17
CustomerIdType (class in smartcloudadmin.enums), 16
CustomerType (class in smartcloudadmin.enums), 16

D

DATA_MIGRATOR (smartcloudadmin.enums.RoleSet attribute), 17

delete() (smartcloudadmin.models.organization.Organization method), 6
delete() (smartcloudadmin.models.subscriber.Subscriber method), 9
delete() (smartcloudadmin.models.subscription.Subscription method), 12
DEREGISTER_PENDING (smartcloudadmin.enums.State attribute), 18
DIRECT (smartcloudadmin.enums.CustomerType attribute), 16

E

EASTERN_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
EN_US (smartcloudadmin.enums.LanguagePreference attribute), 17
entitle() (smartcloudadmin.models.subscriber.Subscriber method), 10
EUR (smartcloudadmin.enums.CurrencyType attribute), 16

F

FALSE (smartcloudadmin.enums.BSSBoolean attribute), 15
FEDERATED (smartcloudadmin.enums.SecurityRealm attribute), 17
filter_subscribers() (smartcloudadmin.models.organization.Organization method), 6
filter_subscriptions() (smartcloudadmin.models.organization.Organization method), 6
from_json() (smartcloudadmin.models.address_set.AddressSet class method), 15
from_json() (smartcloudadmin.models.contact.Contact class method), 14
from_json() (smartcloudadmin.models.organization.Organization class method), 7
from_json() (smartcloudadmin.models.seat.Seat class method), 13
from_json() (smartcloudadmin.models.subscriber.Subscriber class method), 10
from_json() (smartcloudadmin.models.subscription.Subscription class method), 12

G

GBP (smartcloudadmin.enums.CurrencyType attribute), 16
get() (smartcloudadmin.models.organization.Organization class method), 7
get() (smartcloudadmin.models.subscriber.Subscriber class method), 10
get() (smartcloudadmin.models.subscription.Subscription class method), 12
get_basic() (smartcloudadmin.models.organization.Organization class method), 7
get_role_list() (smartcloudadmin.models.subscriber.Subscriber method), 10
get_roles() (smartcloudadmin.models.subscriber.Subscriber method), 10
GLOBALCROSSING_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
GLOBALIVE_ID (smartcloudadmin.enums.CustomerIdType attribute), 16

H

HAWAII_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18

I

IBM_CUSTOMER_NUMBER (smartcloudadmin.enums.CustomerIdType attribute), 16
IBM_CUSTOMER_NUMBER_PREV (smartcloudadmin.enums.CustomerIdType attribute), 16
IBM_SITE_NUMBER (smartcloudadmin.enums.CustomerIdType attribute), 16
INR (smartcloudadmin.enums.CurrencyType attribute), 16
INVOICE (smartcloudadmin.enums.PaymentMethodType attribute), 17

J

JPY (smartcloudadmin.enums.CurrencyType attribute), 16

L

LanguagePreference (class in smartcloudadmin.enums), 16

M

MAILING (smartcloudadmin.enums.AddressType attribute), 15
modified_epoch (smartcloudadmin.models.subscription.Subscription attribute), 12
MODIFIED_FEDERATED (smartcloudadmin.enums.SecurityRealm attribute), 17

MOUNTAIN_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
MULTIPURPOSE (smartcloudadmin.enums.AddressType attribute), 15

N

name (smartcloudadmin.models.subscriber.Subscriber attribute), 10
NEWFOUNDLAND_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
NON_FEDERATED (smartcloudadmin.enums.SecurityRealm attribute), 17
NONE (smartcloudadmin.enums.PaymentMethodType attribute), 17
not_provided() (smartcloudadmin.models.address_set.AddressSet class method), 15
NZD (smartcloudadmin.enums.CurrencyType attribute), 16

O

ORGANISATION (smartcloudadmin.enums.PartyType attribute), 17
Organization (class in smartcloudadmin.models.organization), 3

P

PACIFIC_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18
PARTIAL_FEDERATED (smartcloudadmin.enums.SecurityRealm attribute), 17
PartyRollType (class in smartcloudadmin.enums), 17
PartyType (class in smartcloudadmin.enums), 17
PaymentMethodType (class in smartcloudadmin.enums), 17
PENDING (smartcloudadmin.enums.State attribute), 18
PERSON (smartcloudadmin.enums.PartyType attribute), 17
PURCHASE_ORDER (smartcloudadmin.enums.PaymentMethodType attribute), 17

R

REMOVE_PENDING (smartcloudadmin.enums.State attribute), 18
remove_subscriber() (smartcloudadmin.models.organization.Organization method), 7
remove_subscription() (smartcloudadmin.models.organization.Organization method), 7
reset_password() (smartcloudadmin.models.subscriber.Subscriber method), 10
restore() (smartcloudadmin.models.subscriber.Subscriber method), 10
revoke() (smartcloudadmin.models.subscriber.Subscriber method), 10
RoleSet (class in smartcloudadmin.enums), 17

S

SALESFORCE_ACCOUNT_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
SALESFORCE_CONTACT_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
SALESFORCE_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
SALESFORCE_LEAD_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
SALESFORCE_OPPORTUNITY_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
Seat (class in smartcloudadmin.models.seat), 13
SecurityRealm (class in smartcloudadmin.enums), 17
set_one_time_password() (smartcloudadmin.models.subscriber.Subscriber method), 10
set_password() (smartcloudadmin.models.subscriber.Subscriber method), 10
show_as_row() (smartcloudadmin.models.subscriber.Subscriber method), 10
show_as_row() (smartcloudadmin.models.subscription.Subscription method), 12
show_as_summary() (smartcloudadmin.models.subscriber.Subscriber method), 10
show_as_summary() (smartcloudadmin.models.subscription.Subscription method), 12
SIEBEL_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
smartcloudadmin.enums (module), 15
smartcloudadmin.models.address_set (module), 15
smartcloudadmin.models.contact (module), 14
smartcloudadmin.models.organization (module), 3
smartcloudadmin.models.seat (module), 13
smartcloudadmin.models.subscriber (module), 8
smartcloudadmin.models.subscription (module), 11
SOFT_DELETED (smartcloudadmin.enums.State attribute), 18
State (class in smartcloudadmin.enums), 18
STERLING_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
Subscriber (class in smartcloudadmin.models.subscriber), 8

SUBSCRIBER (smartcloudadmin.enums.PartyRollType attribute), 17
subscriber_count (smartcloudadmin.models.organization.Organization attribute), 8
Subscription (class in smartcloudadmin.models.subscription), 11
subscription_count (smartcloudadmin.models.organization.Organization attribute), 8
SUPPORT (smartcloudadmin.enums.RoleSet attribute), 17
suspend() (smartcloudadmin.models.organization.Organization method), 8
suspend() (smartcloudadmin.models.subscriber.Subscriber method), 10
suspend() (smartcloudadmin.models.subscription.Subscription method), 13
SUSPENDED (smartcloudadmin.enums.State attribute), 18

T

TimeZone (class in smartcloudadmin.enums), 18
TMS_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
transfer_seat() (smartcloudadmin.models.subscription.Subscription method), 13
TRUE (smartcloudadmin.enums.BSSBoolean attribute), 15

U

unassign_role() (smartcloudadmin.models.subscriber.Subscriber method), 10
UNICA_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
UNSET (smartcloudadmin.enums.State attribute), 18
unsuspend() (smartcloudadmin.models.organization.Organization method), 8
unsuspend() (smartcloudadmin.models.subscriber.Subscriber method), 10
unsuspend() (smartcloudadmin.models.subscription.Subscription method), 13
UNYTE_CUSTOMER_ID (smartcloudadmin.enums.CustomerIdType attribute), 16
USD (smartcloudadmin.enums.CurrencyType attribute), 16
USER (smartcloudadmin.enums.RoleSet attribute), 17
USER_ACCOUNT_ASSISTANT (smartcloudadmin.enums.RoleSet attribute), 17

V

VSR (smartcloudadmin.enums.RoleSet attribute), 17

W

WON (smartcloudadmin.enums.CurrencyType attribute), 16

Y

YUKON_STANDARD_TIME (smartcloudadmin.enums.TimeZone attribute), 18