

---

# **CTERA SDK for Python Documentation**

**CTERA Networks**

**Oct 07, 2021**



# CONTENTS:

- 1 CTERA for Python** **1**
- 1.1 Documentation . . . . . 1
- 1.2 Installation . . . . . 1
- 1.3 Importing the Library . . . . . 2
- 1.4 Building Documentation . . . . . 2
- 1.5 Testing . . . . . 2
  
- 2 User Guides** **3**
- 2.1 Global File System . . . . . 3
  - 2.1.1 Global Administration . . . . . 3
    - 2.1.1.1 Instantiate a Global Admin object . . . . . 4
    - 2.1.1.2 Setup . . . . . 5
    - 2.1.1.3 Logging in . . . . . 6
    - 2.1.1.4 Navigating . . . . . 6
    - 2.1.1.5 Core Methods . . . . . 7
    - 2.1.1.6 Storage Nodes . . . . . 7
    - 2.1.1.7 Portals . . . . . 9
    - 2.1.1.8 Plans . . . . . 11
    - 2.1.1.9 Configuration Templates . . . . . 14
    - 2.1.1.10 Servers . . . . . 18
    - 2.1.1.11 Antivirus . . . . . 19
    - 2.1.1.12 Users . . . . . 20
    - 2.1.1.13 Directory Services . . . . . 23
    - 2.1.1.14 Devices . . . . . 25
    - 2.1.1.15 Zones . . . . . 28
    - 2.1.1.16 CloudFS . . . . . 30
    - 2.1.1.17 Timezone . . . . . 33
    - 2.1.1.18 SSL Certificate . . . . . 33
    - 2.1.1.19 Logs . . . . . 34
    - 2.1.1.20 Syslog . . . . . 35
  - 2.1.2 End User Portal . . . . . 35
    - 2.1.2.1 Instantiate a Services Portal object . . . . . 35
  - 2.1.3 File Browser . . . . . 36
    - 2.1.3.1 File Browser . . . . . 37
    - 2.1.3.2 Cloud Drive . . . . . 39
    - 2.1.3.3 Backups . . . . . 43
- 2.2 Edge Filer . . . . . 43
  - 2.2.1 Gateway . . . . . 43
    - 2.2.1.1 Instantiate a Gateway object . . . . . 45
  - 2.2.2 File Browser . . . . . 74

2.2.2.1	Obtaining Access to the Gateway’s File System . . . . .	74
2.3	Agent . . . . .	76
2.4	Miscellaneous . . . . .	77
2.4.1	Logging . . . . .	77
2.4.1.1	Redirecting the log to a file . . . . .	77
2.4.1.2	Disabling the Logger . . . . .	77
2.4.1.3	Changing the Log Level . . . . .	77
2.4.2	Formatting . . . . .	77
<b>3</b>	<b>cterasdk package</b>	<b>79</b>
3.1	Subpackages . . . . .	79
3.1.1	cterasdk.client package . . . . .	79
3.1.1.1	Submodules . . . . .	79
3.1.2	cterasdk.common package . . . . .	83
3.1.2.1	Submodules . . . . .	83
3.1.3	cterasdk.convert package . . . . .	88
3.1.3.1	Submodules . . . . .	88
3.1.4	cterasdk.core package . . . . .	89
3.1.4.1	Subpackages . . . . .	89
3.1.4.2	Submodules . . . . .	95
3.1.5	cterasdk.edge package . . . . .	133
3.1.5.1	Subpackages . . . . .	133
3.1.5.2	Submodules . . . . .	136
3.1.6	cterasdk.lib package . . . . .	174
3.1.6.1	Submodules . . . . .	174
3.1.7	cterasdk.object package . . . . .	178
3.1.7.1	Submodules . . . . .	178
3.1.8	cterasdk.transcript package . . . . .	182
3.1.8.1	Submodules . . . . .	182
3.2	Submodules . . . . .	182
3.2.1	cterasdk.config module . . . . .	182
3.2.2	cterasdk.exception module . . . . .	182
<b>4</b>	<b>Indices and tables</b>	<b>185</b>
<b>5</b>	<b>Help Us Improve the Docs &lt;3</b>	<b>187</b>
	<b>Python Module Index</b>	<b>189</b>
	<b>Index</b>	<b>191</b>

## CTERA FOR PYTHON

A Python SDK for integrating with the CTERA Global File System API. Compatible with Python 3.5+.

### 1.1 Documentation

User documentation is available on [Read the Docs](#).

### 1.2 Installation

Installing via `pip`:

```
$ pip install cterasdk
```

If you receive a certificate error, add the following trusted hosts:

```
$ pip install cterasdk --trusted-host pypi.org --trusted-host files.pythonhosted.org #  
↪ [SSL: CERTIFICATE_VERIFY_FAILED]
```

Installation via proxy:

```
$ pip install cterasdk --proxy http://user:password@proxyserver:port # use proxy
```

Install from source:

```
$ git clone https://github.com/ctera/ctera-python-sdk.git
$ cd ctera-python-sdk
$ python setup.py install
```

### 1.3 Importing the Library

After installation, to get started, open a Python console:

```
>>> from cterasdk import *
```

### 1.4 Building Documentation

Documentation can be compiled by running `make html` from the docs folder. After compilation, open `docs/build/html/index.html`.

### 1.5 Testing

We use the `tox` package to run tests in Python 3. To install, use `pip install tox`. Once installed, run `tox` from the root directory.

```
$ tox
```

## 2.1 Global File System

### 2.1.1 Global Administration

#### Table of Contents

- *Global Administration*
  - *Instantiate a Global Admin object*
  - *Setup*
  - *Logging in*
  - *Navigating*
  - *Core Methods*
  - *Storage Nodes*
  - *Portals*
    - \* *Retrieve Portals*
    - \* *Create a Team Portal*
    - \* *Subscribe a Team Portal to a Plan*
    - \* *Apply Provisioning Changes*
    - \* *Delete a Team Portal*
    - \* *Recover a Team Portal*
  - *Plans*
    - \* *Plan Auto Assignment Rules*
  - *Configuration Templates*
    - \* *Template Auto Assignment Rules*
  - *Servers*
  - *Antivirus*
    - \* *Antivirus Servers*

- *Users*
  - \* *Local Users*
  - \* *Domain Users*
  - \* *Fetch Users & Groups*
- *Directory Services*
- *Devices*
  - \* *Generate Activation Codes*
  - \* *Code Snippets*
- *Zones*
  - \* *Retrieve a Zone*
  - \* *Create a Zone*
  - \* *Add Folders to a Zone*
  - \* *Add Devices to a Zone*
  - \* *Delete a Zone*
- *CloudFS*
  - \* *Folder Groups*
  - \* *Cloud Drive Folders*
- *Timezone*
- *SSL Certificate*
- *Logs*
- *Syslog*

### 2.1.1.1 Instantiate a Global Admin object

**class** `cterasdk.object.Portal.GlobalAdmin`(*host, port=None, https=True*)

Main class for Global Admin operations on a Portal

#### Variables

- **portals** (`cterasdk.core.portals.Portals`) – Object holding the Portals Management APIs
- **servers** (`cterasdk.core.servers.Servers`) – Object holding the Servers Management APIs
- **setup** (`cterasdk.core.setup.Setup`) – Object holding the Portal setup APIs
- **ssl** (`cterasdk.core.ssl.SSL`) – Object holding the Portal SSL Certificate APIs
- **startup** (`cterasdk.core.startup.Startup`) – Object holding the Portal startup APIs
- **syslog** (`cterasdk.core.syslog.Syslog`) – Object holding the Portal syslog APIs
- **antivirus** (`cterasdk.core.antivirus.Antivirus`) – Object holding the Portal Antivirus APIs



- **buckets** (`cterasdk.core.buckets.Buckets`) – Object holding the Portal Storage Node APIs

`__init__(host, port=None, https=True)`

#### Parameters

- **host** (*str*) – The fully qualified domain name, hostname or an IPv4 address of the Portal
- **port** (*int, optional*) – Set a custom port number (0 - 65535), If not set defaults to 80 for http and 443 for https
- **https** (*bool, optional*) – Set to True to require HTTPS, defaults to True

```
admin = GlobalAdmin('portal.ctera.com') # will use HTTPS over port 443
```

**Warning:** for any certificate related error, this library will prompt for your consent in order to proceed. to avoid the prompt, you may configure this library to automatically trust the server's certificate, using: `config.http['ssl'] = 'Trust'`

### 2.1.1.2 Setup

`Setup.init_master(name, email, first_name, last_name, password, domain)`

Initialize the CTERA Portal master server.

#### Parameters

- **name** (*str*) – User name for the new user
- **email** (*str*) – E-mail address of the new user
- **first\_name** (*str*) – The first name of the new user
- **last\_name** (*str*) – The last name of the new user
- **password** (*str*) – Password for the new user
- **domain** (*str*) – The domain suffix for CTERA Portal

```
admin.init_master('admin', 'bruce.wayne@we.com', 'Bruce', 'Wayne', 'password!', 'ctera.
→me')
```

`Setup.init_application_server(ipaddr, secret)`

Initialize a CTERA Portal Application Server.

#### Parameters

- **ipaddr** (*str*) – The CTERA Portal master server IP address
- **secret** (*str*) – A password or a PEM-encoded private key

```
"""Connect a secondary Portal server using a password"""
master_ipaddr = '172.31.53.246'
master_password = 'secret'
admin.init_application_server(master_ipaddr, master_password)

"""Connect a secondary Portal server using a private key"""
master_ipaddr = '172.31.53.246'
```

(continues on next page)

(continued from previous page)

```
master_pk = """...PEM-encoded private key..."""
admin.init_application_server(master_ipaddr, master_pk)
```

`Setup.init_replication_server(ipaddr, secret, replicate_from=None)`  
Initialize a CTERA Portal Database Replication Server.

**Parameters**

- **ipaddr** (*str*) – The CTERA Portal master server IP address
- **secret** (*str*) – A password or a PEM-encoded private key
- **replicate\_from** (*str*) – The name of a CTERA Portal server to replicate from

### 2.1.1.3 Logging in

`GlobalAdmin.test()`  
Verification check to ensure the target host is a Portal.

```
admin.test()
```

`GlobalAdmin.login(username, password)`  
Log in

**Parameters**

- **username** (*str*) – User name to log in
- **password** (*str*) – User password

```
admin.login('admin', 'G3neralZ0d!')
```

`GlobalAdmin.logout()`  
Log out

```
admin.logout()
```

`GlobalAdmin.whoami()`  
Return the name of the logged in user.

**Return** `cterasdk.common.object.Object` The session object of the current user

```
admin.whoami()
```

### 2.1.1.4 Navigating

`Portals.browse_global_admin()`  
Browse the Global Admin

```
admin.portals.browse_global_admin()
```

`Portals.browse(tenant)`  
Browse a tenant

**Parameters** **tenant** (*str*) – Name of the tenant to browse

```
admin.portals.browse('portal')
```

### 2.1.1.5 Core Methods

`GlobalAdmin.show(path, use_file_url=False)`  
Print a schema object as a JSON string.

`GlobalAdmin.show_multi(path, paths, use_file_url=False)`  
Print one or more schema objects as a JSON string.

`GlobalAdmin.get(path, params=None, use_file_url=False)`  
Retrieve a schema object as a Python object.

`GlobalAdmin.put(path, value, use_file_url=False)`  
Update a schema object or attribute.

`GlobalAdmin.execute(path, name, param=None, use_file_url=False)`  
Execute a schema object method.

`GlobalAdmin.query(path, param)`

`GlobalAdmin.show_query(path, param)`

### 2.1.1.6 Storage Nodes

`Buckets.get(name, include=None)`  
Get a Bucket

#### Parameters

- **name** (*str*) – Name of the bucket
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

```
bucket = filer.buckets.get('MainStorage')
print(bucket)

bucket = filer.buckets.get('MainStorage', include=['bucket', 'driver'])
print(bucket.name, bucket.bucket, bucket.driver)
```

`Buckets.add(name, bucket, read_only=False, dedicated_to=None)`  
Add a Bucket

#### Parameters

- **name** (*str*) – Name of the bucket
- **bucket** (`cterasdk.core.types.Bucket`) – Storage bucket to add
- **read\_only** (*bool, optional*) – Set bucket to read-delete only, defaults to False
- **dedicated\_to** (*str, optional*) – Name of a tenant, defaults to None

```
"""Add an Amazon S3 bucket called 'mybucket'"""
bucket = portal_types.AmazonS3('mybucket', 'access-key', 'secret-key')
filer.buckets.add('cterabucket', bucket)
```

```
"""Add an Amazon S3 bucket called 'mybucket', dedicated to a tenant called 'mytenant'"""
```

(continues on next page)

(continued from previous page)

```
bucket = portal_types.AmazonS3('mybucket', 'access-key', 'secret-key')
filer.buckets.add('cterabucket', bucket, dedicated_to='mytenant')
```

```
"""Add a bucket in read-delete only mode"""
bucket = portal_types.AmazonS3('mybucket', 'access-key', 'secret-key')
filer.buckets.add('cterabucket', bucket, read_only=True)
```

Buckets.**modify**(*current\_name*, *new\_name=None*, *read\_only=None*, *dedicated\_to=None*)  
Modify a Bucket

**Parameters**

- **current\_name** (*str*) – The current bucket name
- **new\_name** (*str, optional*) – New name
- **read\_only** (*bool, optional*) – Set bucket to read-delete only
- **dedicated** (*bool, optional*) – Dedicate bucket to a tenant
- **dedicated\_to** (*str, optional*) – Tenant name

```
"""Modify an existing bucket, set it to read-delete only and dedicate it to 'mytenant'"""
filer.buckets.modify('MainStorage', read_only=True, dedicated_to='mytenant')
```

Buckets.**list\_buckets**(*include=None*)  
List Buckets.

To retrieve buckets, you must first browse the Global Administration Portal, using: *GlobalAdmin.portals.browse\_global\_admin()*

**Parameters include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']

```
for bucket in filer.buckets.list_buckets():
    print(bucket)
```

Buckets.**delete**(*name*)  
Delete a Bucket

**Parameters name** (*str*) – Name of the bucket

```
filer.buckets.delete('MainStorage')
```

Buckets.**read\_write**(*name*)  
Set bucket to Read Write

**Parameters name** (*str*) – Name of the bucket

```
filer.buckets.read_write('MainStorage')
```

Buckets.**read\_only**(*name*)  
Set bucket to Read Only

**Parameters name** (*str*) – Name of the bucket

```
filer.buckets.read_only('MainStorage')
```

### 2.1.1.7 Portals

#### Retrieve Portals

`Portals.list_tenants(include=None, portal_type=None)`

List tenants.

To retrieve tenants, you must first browse the Global Administration Portal, using: `GlobalAdmin.portals.browse_global_admin()`

##### Parameters

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **portal\_type** (`cterasdk.core.enum.PortalType`) – The Portal type

```

"""List all tenants"""
for tenant in admin.portals.list_tenants():
    print(tenant)

"""List Team Portals. For each tenant, retrieve its creation date, subscription plan and
↪ activation status"""
for tenant in admin.portals.list_tenants(include=['createDate', 'plan', 'activationStatus
↪'], portal_type=portal_enum.PortalType.Team):
    print(tenant)

```

`Portals.tenants(include_deleted=False)`

Get all tenants

**Parameters** `include_deleted` (*bool, optional*) – Include deleted tenants, defaults to False

```

for tenant in admin.portals.tenants():
    print(tenant.name, tenant.usedStorageQuota, tenant.totalStorageQuota)

```

#### Create a Team Portal

`Portals.add(name, display_name=None, billing_id=None, company=None, plan=None, comment=None)`

Add a new tenant

##### Parameters

- **name** (*str*) – Name of the new tenant
- **display\_name** (*str, optional*) – Display Name of the new tenant, defaults to None
- **billing\_id** (*str, optional*) – Billing ID of the new tenant, defaults to None
- **company** (*str, optional*) – Company Name of the new tenant, defaults to None
- **plan** (*str, optional*) – Subscription plan name to assign to the new tenant, defaults to None
- **comment** (*str, optional*) – Assign a comment to the new tenant, defaults to None

**Return str** A relative url path to the Team Portal

```
"""Create a Team Portal"""
admin.portals.add('acme')

"""Create a Team Portal, including a display name, billing id and a company name"""
admin.portals.add('ctera', 'CTERA', 'Tz9YRDSd8LNfaouzr3Db', 'CTERA Networks')

"""Create a Team Portal and assign it to a pre-configured subscription plan"""
admin.portals.add('ctera', plan = 'Default')
```

### Subscribe a Team Portal to a Plan

`Portals.subscribe(tenant, plan)`

Subscribe a tenant to a plan

#### Parameters

- **name** (*str*) – Name of the tenant
- **str, plan** – Name of the subscription plan

```
admin.portals.subscribe('ctera', '10tb')
```

### Apply Provisioning Changes

`Portals.apply_changes(wait=False)`

Apply provisioning changes.

**Parameters** `wait` (*bool, optional*) – Wait for all changes to apply

```
"""Apply Portal Provisioning Changes"""
admin.portals.apply_changes()
admin.portals.apply_changes(wait=True) # wait for all changes to apply

"""Apply User Provisioning Changes"""
admin.users.apply_changes()
admin.users.apply_changes(wait=True) # wait for all changes to apply
```

### Delete a Team Portal

`Portals.delete(name)`

Delete an existing tenant

**Parameters** `name` (*str*) – Name of the tenant to delete

```
admin.portals.delete_tenant('acme')
```

## Recover a Team Portal

`Portals.undelete(name)`

Undelete a previously deleted tenant

**Parameters** `name` (*str*) – Name of the tenant to undelete

```
admin.portals.undelete_tenant('acme')
```

### 2.1.1.8 Plans

`Plans.get(name, include=None)`

Retrieve subscription plan properties

**Parameters**

- **name** (*str*) – Name of the subscription plan
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** The subscription plan, including the requested fields

```
plan = admin.plans.get('good_plan', ['createDate', 'modifiedDate'])
```

`Plans.list_plans(include=None, filters=None)`

List Plans

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[], optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching Plans

**Return type** *cterasdk.lib.iterator.Iterator*

```
"""List plans and include their creation date"""
for plan in admin.plans.list_plans(['createDate']):
    print(plan)
```

`Plans.by_name(names, include=None)`

Get Plans by their names

**Parameters**

- **names** (*list[str], optional*) – List of names of plans
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[cterasdk.core.query.FilterBuilder], optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching Plans

**Return type** *cterasdk.lib.iterator.Iterator*

```

"""List plans 'PlanOne' and 'PlanTwo'; and retrieve the 'modifiedDate', 'uid' and 'isDefault'
↳properties"""
for plan in admin.plans.by_name(['PlanOne', 'PlanTwo'], ['modifiedDate', 'uid',
↳'isDefault']):
    print(plan)

```

`Plans.add(name, retention=None, quotas=None)`

Add a subscription plan

#### Parameters

- **retention** (*dict, optional*) – The data retention policy
- **quotas** (*dict, optional*) – The items included in the plan and their respective quota

```

"""
Retention Policy (portal_enum.PlanRetention):
- All: All Versions
- Hourly: Hourly Versions
- Daily: Daily Versions
- Weekly: Weekly Versions
- Monthly: Monthly Versions
- Quarterly: Quarterly Versions
- Yearly: Yearly Versions
- Deleted: Recycle Bin

Quotas (portal_enum.PlanItem):
- Storage: Storage Quota, in Gigabytes
- EV4: CTERA Edge Filer, Up to 4 TB of Local Cache
- EV8: CTERA Edge Filer, Up to 8 TB of Local Cache
- EV16: CTERA Edge Filer, Up to 16 TB of Local Cache
- EV32: CTERA Edge Filer, Up to 32 TB of Local Cache
- EV64: CTERA Edge Filer, Up to 64 TB of Local Cache
- EV128: CTERA Edge Filer, Up to 128 TB of Local Cache
- WA: Workstation Backup Agent
- SA: Server Agent
- Share: CTERA Drive Share
- Connect: CTERA Drive Connect
"""

"""
Create the 'good_plan' subscription plan:
1) Retention: 7 daily versions, 12 monthly versions
2) Quotas: 10 x EV16, 5 x EV32, 100 x Cloud Drive (Share)
"""

name = 'good_plan'
retention = {portal_enum.PlanRetention.Daily: 7, portal_enum.PlanRetention.Monthly: 12}
quotas = {portal_enum.PlanItem.EV16: 10, portal_enum.PlanItem.EV32: 5, portal_enum.
↳PlanItem.Share: 100}
admin.plans.add(name, retention, quotas)

```

`Plans.modify(name, retention=None, quotas=None, apply_changes=True)`

Modify a subscription plan

#### Parameters



- **retention** (*dict, optional*) – The data retention policy
- **quotas** (*dict, optional*) – The items included in the plan and their respective quota
- **apply\_changes** (*bool, optional*) – Apply provisioning changes immediately

```

"""
Modify 'good_plan' subscription plan:
1) Retention: 30 daily versions, 36 monthly versions
2) Quotas: 20 x EV16, 10 x EV32, 200 x Cloud Drive (Share)
"""

name = 'good_plan'
retention = {portal_enum.PlanRetention.Daily: 30, portal_enum.PlanRetention.Monthly: 36}
quotas = {portal_enum.PlanItem.EV16: 20, portal_enum.PlanItem.EV32: 10, portal_enum.
↳PlanItem.Share: 200}
admin.plans.modify(name, retention, quotas)

```

Plans.**delete**(*name*)

Delete a subscription plan

**Parameters** **username** (*str*) – The name of the subscription plan

```

name = 'good_plan'
admin.plan.delete(name)

```

## Plan Auto Assignment Rules

PlanAutoAssignPolicy.**get\_policy**()

Get plans auto assignment policy

PlanAutoAssignPolicy.**set\_policy**(*rules, apply\_default=None, default=None, apply\_changes=True*)

Set plans auto assignment policy

### Parameters

- **rules** (*list[cterasdk.common.types.PolicyRule]*) – List of policy rules
- **apply\_default** (*bool, optional*) – If no match found, apply default plan. If not passed, the current config will be kept
- **default** (*str, optional*) – Name of a plan to assign if no match found. Ignored unless the `apply_default` is set to `True`
- **apply\_changes** (*bool, optional*) – Apply provisioning changes upon update, defaults to `True`

```

"""Apply the '100GB' plan to all user names that start with 'adm'"""
c1 = portal_types.PlanCriteriaBuilder.username().startswith('adm').build()
r1 = PolicyRule('100GB', c1)

"""Apply the '200GB' plan to all user names that end with 'inc'"""
c2 = portal_types.PlanCriteriaBuilder.username().endswith('inc').build()
r2 = PolicyRule('200GB', c2)

"""Apply the 'Bingo' plan to all user names that contain 'bing'"""
c3 = portal_types.PlanCriteriaBuilder.username().contains('bing').build()

```

(continues on next page)

```

r3 = PolicyRule('Bingo', c3)

"""Apply the 'ABC' plan to 'alice', 'bob' and 'charlie'"""
c4 = portal_types.PlanCriteriaBuilder.username().isoneof(['alice', 'bob', 'charlie']).
↳build()
r4 = PolicyRule('ABC', c4)

"""Apply the '10TB' plan to read write, read only and support administrators"""
roles = [portal_enum.Role.ReadWriteAdmin, portal_enum.Role.ReadOnlyAdmin, portal_enum.
↳Role.Support]
c5 = portal_types.PlanCriteriaBuilder.role().include(roles).build()
r5 = PolicyRule('10TB', c5)

"""Apply the 'TechStaff' plan to the 'Product' and 'Support' groups"""
c6 = portal_types.PlanCriteriaBuilder.user_groups().include(['Product', 'Support']).
↳build()
r6 = PolicyRule('TechStaff', c6)

admin.plans.auto_assign.set_policy([r1, r2, r3, r4, r5, r6])

"""Remove all policy rules"""
admin.plans.auto_assign.set_policy([])

"""Do not assign a default plan if no match applies"""
admin.plans.auto_assign.set_policy([], False)

"""Assign 'Default' if no match applies"""
admin.plans.auto_assign.set_policy([], True, 'Default')

```

### 2.1.1.9 Configuration Templates

Templates.**get**(*name*, *include=None*)  
Get a Configuration Template

#### Parameters

- **name** (*str*) – Name of the template
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

```
admin.templates.get('MyTemplate')
```

Templates.**list\_templates**(*include=None*, *filters=None*)  
List Configuration Templates.

To retrieve templates, you must first browse the tenant, using: *GlobalAdmin.portals.browse()*

#### Parameters

- **include** (*list[str]*, *optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[]*, *optional*) – List of additional filters, defaults to None

```

for template in admin.templates.list_templates(include=['name', 'description',
↳ 'modifiedDate']):
    print(template.name, template.description, template.modifiedDate)

```

Templates.**add**(*name, description=None, include\_sets=None, exclude\_sets=None, apps=None, backup\_schedule=None, versions=None, scripts=None, cli\_commands=None*)

Add a Configuration Template

#### Parameters

- **name** (*str*) – Name of the template
- **description** (*str, optional*) – Template description
- **include\_sets** (*list[cterasdk.common.types.FilterBackupSet], optional*) – List of backup sets to include
- **exclude\_sets** (*list[cterasdk.common.types.FilterBackupSet], optional*) – List of backup sets to exclude
- **apps** (*list[cterasdk.common.enum.Application], optional*) – List of applications to back up
- **backup\_schedule** (*cterasdk.common.types.TaskSchedule, optional*) – Backup schedule
- **versions** (*list[cterasdk.core.types.PlatformVersion], optional*) – List of platforms and their associated versions. Pass *None* to inherit the default settings from the Global Administration Portal
- **scripts** (*list[cterasdk.core.types.TemplateScript], optional*) – Scripts to execute after logon, before or after backup
- **cli\_commands** (*list[str], optional*) – Template CLI commands to execute

This library provides several classes, methods and enumerators to assist in creating configuration templates:

- #. Builder class for filtered backup sets. *cterasdk.common.types.FileFilterBuilder*
- #. A class representing a backup include or exclude set. *cterasdk.common.types.FilterBackupSet*
- #. Builder class for defining backup schedule. *cterasdk.common.types.BackupScheduleBuilder*
- #. A time-range class, used to configure backups to run at a specific time. *cterasdk.common.types.TimeRange*
- #. Enumerator containing applications supported for backup. *cterasdk.common.enum.Application*
- #. A named tuple defining a platform and a software version. *cterasdk.core.types.PlatformVersion*
- #. Enumerator containing a list of platforms. *cterasdk.core.enum.Platform*

```

"""Include all 'pptx', 'xlsx' and 'docx' file types for all users"""
docs = common_types.FileFilterBuilder.extensions().include(['pptx', 'xlsx', 'docx']).
↳ build()
include_sets = common_types.FilterBackupSet('Documents', filter_rules=[docs],
↳ EnvironmentVariables.ALLUSERSPROFILE)
    template_dirs=[portal_enum.

"""Exclude all 'cmd', 'exe' and 'bat' file types for all users"""
programs = common_types.FileFilterBuilder.extensions().include(['cmd', 'exe', 'bat']).
↳ build()
exclude_sets = common_types.FilterBackupSet('Programs', filter_rules=[programs],
↳ EnvironmentVariables.ALLUSERSPROFILE)
    template_dirs=[portal_enum.

```

(continues on next page)

(continued from previous page)

```

"""Schedule backup to run periodically"""
backup_schedule = common_types.BackupScheduleBuilder.interval(hours=6) # periodically,
↳ every 6 hours
backup_schedule = common_types.BackupScheduleBuilder.interval(hours=0, minutes=30) #
↳ periodically, every 30 minutes

"""Schedule backup for a specific time"""
time_range = common_types.TimeRange().start('07:00:00').days(common_enum.DayOfWeek.
↳ Weekdays).build() # 7am, on weekdays
backup_schedule = common_types.BackupScheduleBuilder.window(time_range)

"""Backup applications"""
apps = [common_enum.Application.NTDS, common_enum.Application.HyperV] # back up Active,
↳ Directory and Hyper-V
apps = common_enum.Application.All # back up all applications

"""Configure software versions"""
versions = [portal_types.PlatformVersion(portal_enum.Platform.Edge_7, '7.0.981.7')] #
↳ use 7.0.981.7 for v7 Edge Filers

"""Configure Scripts"""
scripts = [
    portal_types.TemplateScript.windows().after_logon('echo Current directory: %cd%'),
    portal_types.TemplateScript.linux().before_backup('./mysqldump -u admin website > /
↳ mnt/backup/backup.sql'),
    portal_types.TemplateScript.linux().after_backup('rm /mnt/backup/backup.sql')
]

"""Configure CLI Commands"""
cli_commands = [
    'set /config/agent/stubs/deleteFilesOfCachedFolderOnDisable false',
    'add /config/agent/stubs/allowedExplorerExtensions url'
]

admin.templates.add('MyTemplate', 'woohoo', include_sets=[include_sets], exclude_
↳ sets=[exclude_sets],
    backup_schedule=backup_schedule, apps=apps, versions=versions,
↳ scripts=scripts, cli_commands=cli_commands)

```

Templates.**set\_default**(name, wait=False)

Set a Configuration Template as the default template

#### Parameters

- **name** (str) – Name of the template
- **wait** (bool, optional) – Wait for all changes to apply, defaults to *False*

```

admin.templates.set_default('MyTemplate')

admin.templates.set_default('MyTemplate', wait=True) # wait for template changes to,
↳ apply

```

Templates.**remove\_default**(*name, wait=False*)

Set a Configuration Template not to be the default template

#### Parameters

- **name** (*str*) – Name of the template
- **wait** (*bool, optional*) – Wait for all changes to apply, defaults to *False*

```
admin.templates.remove_default('MyTemplate')
```

```
admin.templates.remove_default('MyTemplate', wait=True) # wait for template changes to
↪ apply
```

TemplateAutoAssignPolicy.**apply\_changes**(*wait=False*)

Apply provisioning changes.

**Parameters** **wait** (*bool, optional*) – Wait for all changes to apply, defaults to *False*

```
admin.templates.auto_assign.apply_changes()
```

```
admin.templates.auto_assign.apply_changes(wait=True) # wait for template changes to
↪ apply
```

## Template Auto Assignment Rules

TemplateAutoAssignPolicy.**get\_policy**()

Get templates auto assignment policy

TemplateAutoAssignPolicy.**set\_policy**(*rules, apply\_default=None, default=None, apply\_changes=True*)

Set templates auto assignment policy

#### Parameters

- **rules** (*list[cterasdk.common.types.PolicyRule]*) – List of policy rules
- **apply\_default** (*bool, optional*) – If no match found, apply default template. If not passed, the current config will be kept
- **default** (*str, optional*) – Name of a template to assign if no match found. Ignored unless the **apply\_default** is set to **True**
- **apply\_changes** (*bool, optional*) – Apply changes upon update, defaults to **True**

```
"""Apply the 'ESeries' template to devices of type: C200, C400, C800, C800P"""
device_types = [portal_enum.DeviceType.C200, portal_enum.DeviceType.C400, portal_enum.
↪ DeviceType.C800, portal_enum.DeviceType.C800P]
c1 = portal_types.TemplateCriteriaBuilder.type().include(device_types).build()
r1 = PolicyRule('ESeries', c1)

"""Apply the 'Windows' template to devices that use a 'Windows' operating system"""
c2 = portal_types.TemplateCriteriaBuilder.os().contains('Windows').build()
r2 = PolicyRule('Windows', c2)

"""Apply the 'CTERA7' template to devices running version 7"""
c3 = portal_types.TemplateCriteriaBuilder.version().startswith('7.0').build()
r3 = PolicyRule('CTERA7', c3)
```

(continues on next page)

(continued from previous page)

```

"""Apply the 'WD5' template to devices that their hostname ends with 'WD5'"""
c4 = portal_types.TemplateCriteriaBuilder.hostname().endswith('WD5').build()
r4 = PolicyRule('WD5', c4)

"""Apply the 'Beta' template to devices that their name is one of"""
c5 = portal_types.TemplateCriteriaBuilder.name().isoneof(['DEV1', 'DEV2', 'DEV3']).
↳ build()
r5 = PolicyRule('Beta', c5)

admin.templates.auto_assign.set_policy([r1, r2, r3, r4, r5])

"""Remove all policy rules"""
admin.templates.auto_assign.set_policy([])

"""Do not assign a default template if no match applies"""
admin.templates.auto_assign.set_policy([], False)

"""Assign 'Default' if no match applies"""
admin.templates.auto_assign.set_policy([], True, 'Default')

```

### 2.1.1.10 Servers

`Servers.get(name, include=None)`

Retrieve server properties

#### Parameters

- **name** (*str*) – Name of the server
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** The server, including the requested fields

```

"""Retrieve a server"""
server = admin.servers.get('server', ['isApplicationServer', 'renderingServer'])
print(server.isApplicationServer, server.renderingServer)

```

`Servers.list_servers(include=None)`

Retrieve the servers that comprise CTERA Portal.

To retrieve servers, you must first browse the Global Administration Portal, using: `GlobalAdmin.portals.browse_global_admin()`

**Parameters include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']

```

"""Retrieve all servers"""
servers = admin.servers.list_servers() # will only retrieve the server name
for server in servers:
    print(server.name)

"""Retrieve multiple server attributes"""
servers = admin.servers.list_servers(include = ['name', 'connected', 'isApplicationServer
↳ ', 'mainDB'])

```

(continues on next page)

(continued from previous page)

```
for server in servers:
    print(server)
```

`Servers.modify(name, server_name=None, app=None, preview=None, enable_public_ip=None, public_ip=None, allow_user_login=None, enable_replication=None, replica_of=None)`

Modify a Portal server

#### Parameters

- **name** (*str*) – The current server name
- **server\_name** (*str, optional*) – New server name
- **app** (*bool, optional*) – Application server
- **preview** (*bool, optional*) – Preview server
- **enable\_public\_ip** (*bool, optional*) – Enable or disable public NAT address
- **public\_ip** (*str, optional*) – Public NAT address
- **allow\_user\_login** (*bool, optional*) – Allow or disallow logins to this server
- **enable\_replication** (*bool, optional*) – Enable or disable database replication
- **replica\_of** (*str, optional*) – Configure as a replicate of another Portal server. *enable\_replication* must be set to *True*

```
admin.servers.modify('server2', server_name='replica', app=False, enable_
↪replication=True, replica_of='maindb') # rename and enable database replication

admin.servers.modify('server2', allow_user_login=False) # disable logins to this server

admin.servers.modify('server2', enable_public_ip=True, public_ip='33.191.55.2') #
↪configure a public NAT ip address
```

### 2.1.1.11 Antivirus

`Antivirus.list_servers(include=None)`

List the antivirus servers

**Parameters include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'type']

`Antivirus.status()`

Get antivirus service status

`Antivirus.rescan()`

Scan all files using the latest antivirus update. This may take a while

`Antivirus.suspend()`

Suspend antivirus scanning

`Antivirus.unsuspend()`

Unsuspend antivirus scanning

## Antivirus Servers

`AntivirusServers.get(name)`

Get an antivirus server's configuration

**Parameters** `name` (*str*) – Server name

`AntivirusServers.add(name, vendor, url, connection_timeout=5)`

Add an antivirus server

**Parameters**

- **name** (*str*) – Server name
- **vendor** (`cterasdk.core.enum.AntivirusType`) – Server type
- **url** (*str*) – Server URL (example: `http://your-antivirus.server.local:1234/signature`)
- **connection\_timeout** (*int, optional*) – Server connection timeout (in seconds), defaults to 5 seconds

`AntivirusServers.delete(name)`

Remove an antivirus server

`AntivirusServers.suspend(name)`

Suspend an antivirus server

`AntivirusServers.unsuspend(name)`

Unsuspend antivirus scanning

### 2.1.1.12 Users

`Users.delete(user)`

Delete a user

**Parameters** `user` (`cterasdk.core.types.UserAccount`) – the user account

```
"""Delete a local user"""
alice = portal_types.UserAccount('alice')
admin.users.delete(alice)

"""Delete a domain user"""
bruce = portal_types.UserAccount('bruce', 'domain.ctera.local')
admin.users.delete(bruce)
```



## Local Users

`Users.list_local_users(include=None)`

List all local users

**Parameters** `include` (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** Iterator for all local users

**Return type** `cterasdk.lib.iterator.Iterator`

```
users = admin.users.list_local_users()

for user in users:
    print(user.name)

users = admin.users.list_local_users(include = ['name', 'email', 'firstName', 'lastName', ''])

for user in users:
    print(user)
```

`Users.add(name, email, first_name, last_name, password, role, company=None, comment=None, password_change=False)`

Create a local user account

### Parameters

- **name** (*str*) – User name for the new user
- **email** (*str*) – E-mail address of the new user
- **first\_name** (*str*) – The first name of the new user
- **last\_name** (*str*) – The last name of the new user
- **password** (*str*) – Password for the new user
- **role** (`cterasdk.core.enum.Role`) – User role of the new user
- **company** (*str, optional*) – The name of the company of the new user, defaults to None
- **comment** (*str, optional*) – Additional comment for the new user, defaults to None
- **password\_change** (*variable, optional*) – Require the user to change the password on the first login. Pass `datetime.date` for a specific date, integer for days from creation, or `True` for immediate, defaults to `False`

```
"""Create a local user"""
admin.users.add('bruce', 'bruce.wayne@we.com', 'Bruce', 'Wayne', 'G0th4amCity!')
```

`Users.modify(current_username, new_username=None, email=None, first_name=None, last_name=None, password=None, role=None, company=None, comment=None)`

Modify a local user account

### Parameters

- **current\_username** (*str*) – The current user name
- **new\_username** (*str, optional*) – New name

- **email** (*str, optional*) – E-mail address
- **first\_name** (*str, optional*) – First name
- **last\_name** (*str, optional*) – Last name
- **password** (*str, optional*) – Password
- **role** (`cterasdk.core.enum.Role`, *optional*) – User role
- **company** (*str, optional*) – Company name
- **comment** (*str, optional*) – Comment

```
"""Modify a local user"""
admin.users.modify('bruce', 'bwayne@we.com', 'Bruce', 'Wayne', 'Str0ngP@ssword!', 'Wayne_
↳Enterprises')
```

### Domain Users

`Users.list_domains()`

List all domains

**Return list** List of all domains

`Users.list_domain_users(domain, include=None)`

List all the users in the domain

**Parameters include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** Iterator for all the domain users

**Return type** `cterasdk.lib.iterator.Iterator`

```
users = admin.users.list_domain_users('domain.ctera.local') # will only retrieve the 'name
↳ attribute
for user in users:
    print(user.name)

"""Retrieve additional user attributes"""
users = admin.users.list_domain_users('domain.ctera.local', include = ['name', 'email',
↳ 'firstName', 'lastName'])
print(user)
```

### Fetch Users & Groups

`DirectoryService.fetch(active_directory_accounts)`

Instruct the Portal to fetch the provided Active Directory Accounts

**Parameters active\_directory\_accounts** (*list[cterasdk.core.types.PortalAccount]*)  
– List of Active Directory Accounts to fetch

**Returns** Response Code

```
"""Fetch domain users"""

alice = portal_types.UserAccount('alice', 'domain.ctera.local')
```

(continues on next page)

(continued from previous page)

```
bruce = portal_types.UserAccount('bruce', 'domain.ctera.local')
admin.directoryservice.fetch([alice, bruce])
```

### 2.1.1.13 Directory Services

`DirectoryService.connect`(*domain, username, password, directory='ActiveDirectory', domain\_controllers=None, ou=None, ssl=False, krb=False, mapping=None, acl=None, default='Disabled', fetch='Lazy'*)

Connect a Portal tenant to directory services

#### Parameters

- **domain** (*str*) – The directory services domain to connect to
- **username** (*str*) – The user name to use when connecting to the active directory services
- **password** (*str*) – The password to use when connecting to the active directory services
- **ou** (*str, optional*) – The OU path to use when connecting to the active directory services, defaults to *None*
- **directory** (`cterasdk.core.enum.DirectoryServiceType`, *optional*) – The directory service type, defaults to *ActiveDirectory*
- **domain\_controllers** (`cterasdk.core.types.DomainControllers`, *optional*) – Connect to a primary and a secondary domain controllers, defaults to *None*
- **ssl** (*bool, optional*) – Connect using SSL, defaults to *False*
- **krb** (*bool, optional*) – Connect using Kerberos, defaults to *False*
- **list**[`cterasdk.core.types.ADDomainIDMapping`], *optional* – The directory services UID/GID mapping
- **acl** (*list*[`cterasdk.core.types.AccessControlEntry`], *optional*) – List of access control entries and their associated roles
- **default** (`cterasdk.core.enum.Role`) – Default role if no match applies, defaults to *None*
- **fetch** (*str, optional*) – Configure identity fetching mode, defaults to *Lazy*

```
"""Connect to Active Directory using a primary domain controller, configure domain UID/
↳GID mapping and access control"""
mapping = [portal_types.ADDomainIDMapping('demo.local', 2000001, 50000000), portal_types.
↳ADDomainIDMapping('trusted.local', 50000001, 100000000)]
rw_admin_group = portal_types.AccessControlEntry(
    portal_types.GroupAccount('ctera_admins', 'demo.local'),
    portal_enum.Role.ReadWriteAdmin
)
ro_admin_user = portal_types.AccessControlEntry(
    portal_types.UserAccount('jsmith', 'demo.local'),
    portal_enum.Role.ReadOnlyAdmin
)
admin.directoryservice.connect('demo.local', 'svc_account', 'P@ssw@rd1', mapping=mapping,
↳ domain_controllers=portal_types.DomainControllers('172.54.3.52'), acl=[rw_admin, ro_
↳admin])
```

DirectoryService.**domains()**

Get domains

**Return list(str)** List of names of all discovered domains

```
print(admin.directoryservice.domains())
```

DirectoryService.**get\_connected\_domain()**

Get the connected domain information. Returns *None* if the Portal tenant is not connected to a domain

**Return str** The connected domain

```
print(admin.directoryservice.get_connected_domain())
```

DirectoryService.**get\_advanced\_mapping()**

Retrieve directory services advanced mapping configuration

**Returns** A dictionary of domain mapping objects

**Return type** dict

```
for domain, mapping in admin.directoryservice.get_advanced_mapping().items():
    print(domain, mapping)
```

DirectoryService.**set\_advanced\_mapping(mapping)**

Configure advanced mapping

**Parameters mapping** (*list[cterasdk.core.types.ADDomainIDMapping]*) – The directory services UID/GID mapping

```
"""Configure UID/GID mapping"""
mapping = [portal_types.ADDomainIDMapping('demo.local', 2000001, 50000000), portal_types.
↪ADDomainIDMapping('trusted.local', 50000001, 100000000)]
admin.directoryservice.set_advanced_mapping(mapping)
```

DirectoryService.**get\_access\_control()**

Retrieve directory services access control list

**Returns** List of access control entries

**Return type** list[cterasdk.core.types.AccessControlEntry]

```
for ace in admin.directoryservice.get_access_control():
    print(ace.account, ace.role)
```

DirectoryService.**set\_access\_control(acl=None, default=None)**

Configure directory services access control

**Parameters**

- **acl** (*list[cterasdk.core.types.AccessControlEntry]*, *optional*) – List of access control entries and their associated roles
- **default** (*cterasdk.core.enum.Role*) – Default role if no match applies, defaults to *None*

```
"""Configure access control for a domain group and a domain user. Set the default role.
↪to 'Disabled'"""
rw_admin_group = portal_types.AccessControlEntry(
    portal_types.GroupAccount('ctera_admins', 'demo.local'),
```

(continues on next page)

(continued from previous page)

```

portal_enum.Role.ReadWriteAdmin
)
ro_admin_user = portal_types.AccessControlEntry(
    portal_types.UserAccount('jsmith', 'demo.local'),
    portal_enum.Role.ReadOnlyAdmin
)
admin.directoryservice.set_access_control([rw_admin_group, ro_admin_user], portal_enum.
↪Role.Disabled)

```

**DirectoryService.get\_default\_role()**

Retrieve the default role assigned when no access control entry match was found

```
print(admin.directoryservice.get_default_role())
```

**DirectoryService.disconnect()**

Disconnect a Portal tenant from directory services

```
admin.directoryservice.disconnect()
```

**2.1.1.14 Devices****Devices.device(device\_name, tenant=None, include=None)**

Get a Device by its name

**Parameters**

- **device\_name** (*str*) – Name of the device to retrieve
- **tenant** (*str, optional*) – Tenant of the device, defaults to the tenant in the current session
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']

**Returns** Managed Device**Return type** ctera.object.Gateway.Gateway or ctera.object.Agent.Agent**Devices.filers(include=None, allPortals=False, deviceTypes=None)**

Get Filers

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False
- **deviceTypes** (*list[cterasdk.core.enum.DeviceType.Gateways]*) – Types of Filers, defaults to all Filer types

**Returns** Iterator for all matching Filers**Return type** cterasdk.lib.iterator.Iterator[cterasdk.object.Gateway.Gateway]

```
"""Retrieve all Gateways from the current tenant"""
```

```
filers = admin.devices.filers()
```

(continues on next page)

```

for filer in filers:
    print(filer.name) # will print the Gateway name

    """Retrieve additional Gateway attributes"""

filers = admin.devices.filers(['owner', 'deviceConnectionStatus'])

    """Retrieve nested attributes using the '.' delimiter"""

filers = admin.devices.filers(['deviceReportedStatus.status.device.runningFirmware'])

    """Retrieve filers from all portals"""

admin.portals.browse_global_admin()

filers = admin.devices.filers(allPortals = True)

    """Retrieve C200's and C400's from all portals"""

admin.portals.browse_global_admin()

filers = admin.devices.filers(allPortals = True, deviceTypes = ['C200', 'C400'])

```

Devices.**agents**(include=None, allPortals=False)

Get Agents

**Parameters**

- **include** (list[str], optional) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (bool, optional) – Search in all portals, defaults to False

**Returns** Iterator for all matching Agents

**Return type** *cterasdk.lib.iterator.Iterator[cterasdk.object.Agent.Agent]*

```

    """Retrieve all Agents from the current tenant"""

agents = admin.devices.agents()

for agent in agents:
    print(agent.name) # will print the Agent name

    """Retrieve all Agents and the underlying OS name"""

agents = admin.devices.agents(['deviceReportedStatus.status.agent.details.osName'])

```

Devices.**servers**(include=None, allPortals=False)

Get Servers

**Parameters**

- **include** (list[str], optional) – List of fields to retrieve, defaults to ['name', 'portal',

‘deviceType’]

- **allPortals** (*bool, optional*) – Search in all portals, defaults to False

**Returns** Iterator for all matching Servers

**Return type** *cterasdk.lib.iterator.Iterator*

```
server_agents = admin.devices.server()
```

Devices.**desktops**(*include=None, allPortals=False*)

Get Desktops

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to [‘name’, ‘portal’, ‘deviceType’]
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False

**Returns** Iterator for all matching Desktops

**Return type** *cterasdk.lib.iterator.Iterator*

```
desktop_agents = admin.devices.desktop_agents()
```

Devices.**by\_name**(*names, include=None*)

Get Devices by their names

**Parameters**

- **names** (*list[str], optional*) – List of names of devices
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to [‘name’, ‘portal’, ‘deviceType’]

**Returns** Iterator for all matching Devices

**Return type** *cterasdk.lib.iterator.Iterator*

## Generate Activation Codes

Activation.**generate\_code**(*username=None, tenant=None*)

Generate device activation code

**Parameters**

- **username** (*str, optional*) – User name used for activation, defaults to None
- **tenant** (*str, optional*) – Tenant name used for activation, defaults to None

**Returns** Portal Activation Code

**Return type** *str*

```
"""Generate a device activation code"""
code = admin.activation.generate_code() # will generate a code for the current, logged_
↳ on, user
code = admin.activation.generate_code('bruce') # will generate a code for 'bruce' in the_
↳ current tenant
```

(continues on next page)

(continued from previous page)

```
code = admin.activation.generate_code('batman', 'gotham') # will generate a code for  
↳ 'bruce' in the 'gotham' tenant
```

---

**Note:** Read Write Administrator, granted with the “Super User” role permission, can generate 200 codes every 5 minutes

---

## Code Snippets

Generate activation codes for all domain users

```
# ... login ...  
  
users = admin.users.list_domain_users('dc.ctera.local') # obtain a list of domain users  
  
for user in users:  
    activation_code = admin.activation.generate_code(user.name) # generate activation_  
↳ code  
    print((user.name, activation_code))  
  
# ... logout ...
```

### 2.1.1.15 Zones

To manage zones, you must be a Read Write Administrator

#### Retrieve a Zone

`Zones.get(name)`

Get zone by name

**Parameters** `name` (*str*) – The name of the zone to get

**Returns** The requested zone

```
zone = admin.zones.get('ZN-001')
```



## Create a Zone

`Zones.add(name, policy_type='selectedFolders', description=None)`

Add a new zone

### Parameters

- **name** (*str*) – The name of the new zone
- **policy\_type** (`cterasdk.core.enum.PolicyType`, *optional*) – Policy type of the new zone, defaults to `cterasdk.core.enum.PolicyType.SELECT`
- **description** (*str*, *optional*) – The description of the new zone

```

"""
Policy Types:
- All: Include all cloud folders
- Select: Select one or more cloud folders to include
- None: Create an empty zone
"""

"""Create a zone with a description"""
admin.zones.add('ZN-NYS-001', description = 'The New York State Zone')

"""Create a zone and include all folders"""
admin.zones.add('ZN-NYS-002', 'All', 'All Folders')

"""Create an empty zone"""
admin.zones.add('ZN-NYS-003', 'None', 'Empty Zone')

```

## Add Folders to a Zone

`Zones.add_folders(name, folder_finding_helpers)`

Add the folders to the zone

### Parameters

- **name** (*str*) – The name of the zone
- **folder\_finding\_helpers** (*list*[`cterasdk.core.types.CloudFSFolderFindingHelper`]) – List of folder names and owners

```

"""
Add the following cloud folders to zone: 'ZN-001'

1) 'Accounting' folder owned by 'Bruce'
2) 'HR' folder owned by 'Diana'
"""

accounting = portal_types.CloudFSFolderFindingHelper('Accounting', 'Bruce')
hr = portal_types.CloudFSFolderFindingHelper('HR', 'Diana')

admin.zones.add_folders('ZN-001', [accounting, hr])

```

## Add Devices to a Zone

Zones.**add\_devices**(*name*, *device\_names*)

Add devices to a zone

### Parameters

- **name** (*str*) – The name of the zone to add devices to
- **device\_names** (*list[str]*) – The names of the devices to add to the zone

```
admin.zones.add_devices('ZN-001', ['vGateway-01ba', 'vGateway-bd02'])
```

## Delete a Zone

Zones.**delete**(*name*)

Delete a zone

**Parameters** **name** (*str*) – The name of the zone to delete

```
admin.zones.delete('ZN-001')
```

### 2.1.1.16 CloudFS

To manage the Cloud File System, folder groups, backup and cloud drive folders, you must be a Read Write Administrator

## Folder Groups

CloudFS.**list\_folder\_groups**(*include=None*, *user=None*)

List folder groups

### Parameters

- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'owner']
- **user** (*cterasdk.core.types.UserAccount*) – User account of the folder group owner

**Returns** Iterator for all folder groups

```
"""List all folder groups"""
folder_groups = admin.cloudfs.list_folder_groups()
for folder_group in folder_groups:
    print(folder_group.name, folder_group.owner)

"""List folder groups owned by a domain user"""
bruce = portal_types.UserAccount('bruce', 'domain.ctera.local')
folder_groups = admin.cloudfs.list_folder_groups(user=bruce)
```

CloudFS.**mkfg**(*name*, *user=None*, *deduplication\_method\_type=None*)

Create a new Folder Group

### Parameters

- **name** (*str*) – Name of the new folder group

- **user** (`cterasdk.core.types.UserAccount`) – User account, the user directory and name of the new folder group owner (default to None)
- **deduplication\_method\_type** (`cterasdk.core.enum.DeduplicationMethodType`) – Deduplication-Method

```

"""Create a Folder Group, owned by a local user account 'svc_account'"""
svc_account = portal_types.UserAccount('svc_account')
admin.cloudfs.mkfg('FG-001', svc_account)

"""Create a Folder Group, owned by the domain user 'ctera.local\wbruce'"""
wbruce = portal_types.UserAccount('wbruce', 'ctera.local')
admin.cloudfs.mkfg('FG-002', wbruce)

admin.cloudfs.mkfg('FG-003') # without an owner

```

CloudFS.**rmfg**(*name*)

Remove a Folder Group

**Parameters** *name* (*str*) – Name of the folder group to remove

```
admin.cloudfs.rmfg('FG-001')
```

## Cloud Drive Folders

CloudFS.**list\_folders**(*include=None, list\_filter='NonDeleted, user=None*)

List Cloud Drive folders.

### Parameters

- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'group', 'owner']
- **list\_filter** (`cterasdk.core.enum.ListFilter`) – Filter the list of Cloud Drive folders, defaults to non-deleted folders
- **user** (`cterasdk.core.types.UserAccount`) – User account of the cloud folder owner

**Returns** Iterator for all Cloud Drive folders

```

"""List all cloud drive folders"""
cloud_drive_folders = admin.cloudfs.list_folders()
for cloud_drive_folder in cloud_drive_folders:
    print(cloud_drive_folder)

"""List cloud drive folders owned by a domain user"""
bruce = portal_types.UserAccount('bruce', 'domain.ctera.local')
cloud_drive_folders = admin.cloudfs.list_folders(user=bruce)

"""List both deleted and non-deleted cloud drive folders"""
cloud_drive_folders = admin.cloudfs.list_folders(list_filter=portal_enum.ListFilter.All)

"""List deleted cloud drive folders"""
cloud_drive_folders = admin.cloudfs.list_folders(list_filter=portal_enum.ListFilter.
↳ Deleted)

```

CloudFS.**mkdir**(*name, group, owner, winacIs=True, description=None*)

Create a new directory

**Parameters**

- **name** (*str*) – Name of the new directory
- **group** (*str*) – The Folder Group to which the directory belongs
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the new directory
- **winacls** (*bool, optional*) – Use Windows ACLs, defaults to True
- **description** (*str, optional*) – Cloud drive folder description

```

"""Create a Cloud Drive folder, owned by a local user account 'svc_account'"""
svc_account = portal_types.UserAccount('svc_account')
admin.cloudfs.mkdir('DIR-001', 'FG-001', svc_account)
admin.cloudfs.mkdir('DIR-003', 'FG-003', svc_account, winacls = False) # disable Windows_
↔ACL's

"""Create a Cloud Drive folder, owned by the domain user 'ctera.local\wbruce'"""
wbruce = portal_types.UserAccount('wbruce', 'ctera.local')
admin.cloudfs.mkdir('DIR-002', 'FG-002', wbruce)

```

`CloudFS.delete(name, owner)`  
Delete a Cloud Drive Folder

**Parameters**

- **name** (*str*) – Name of the Cloud Drive Folder to delete
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the Cloud Drive Folder to delete

```

"""Delete a Cloud Drive folder, owned by the local user account 'svc_account'"""
svc_account = portal_types.UserAccount('svc_account')
admin.cloudfs.delete('DIR-001', svc_account)

"""Delete a Cloud Drive folder, owned by the domain user 'ctera.local\wbruce'"""
wbruce = portal_types.UserAccount('wbruce', 'ctera.local')
admin.cloudfs.delete('DIR-002', wbruce)

```

`CloudFS.undelete(name, owner)`  
Un-Delete a Cloud Drive Folder

**Parameters**

- **name** (*str*) – Name of the Cloud Drive Folder to un-delete
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the Cloud Drive Folder to delete

```

"""Recover a deleted Cloud Drive folder, owned by the local user account 'svc_account'"""
svc_account = portal_types.UserAccount('svc_account')
admin.cloudfs.undelete('DIR-001', svc_account)

"""Recover a deleted Cloud Drive folder, owned by the domain user 'ctera.local\wbruce'"""
wbruce = portal_types.UserAccount('wbruce', 'ctera.local')
admin.cloudfs.undelete('DIR-002', wbruce)

```

### 2.1.1.17 Timezone

`GlobalSettings.get_timezone()`  
Get timezone

```
admin.settings.global_settings.get_timezone()
```

`GlobalSettings.set_timezone(timezone)`  
Set timezone

**Parameters** `timezone` (*str*) – Timezone

```
admin.settings.global_settings.set_timezone('(GMT-05:00) Eastern Time (US , Canada)')
```

### 2.1.1.18 SSL Certificate

`SSL.get()`  
Retrieve details of the current installed SSL certificate

**Return** `cterasdk.common.object.Object` An object including the SSL certificate details

```
certificate = admin.ssl.get()
print(certificate)
```

`SSL.thumbprint()`  
Get the SHA1 thumbprint of the Portal SSL certificate

```
print(admin.ssl.thumbprint)
```

`SSL.export(destination=None)`  
Export the Portal SSL Certificate to a ZIP archive

**Parameters** `destination` (*str, optional*) – File destination, defaults to the default directory

```
admin.ssl.export()
```

```
admin.ssl.export(r'C:\Temp') # export to an alternate location
```

`SSL.import_from_zip(zipfile)`  
Import an SSL Certificate to CTERA Portal from a ZIP archive

**Parameters** `zipfile` (*str*) – A zip archive including the private key and SSL certificate chain

```
admin.ssl.import_from_zip(r'C:\Users\jsmith\Downloads\certificate.zip')
```

`SSL.import_from_chain(private_key, *certificates)`  
Import an SSL Certificate to CTERA Portal from a chain

**Parameters**

- **private\_key** (*str*) – The PEM-encoded private key, or a path to the PEM-encoded private key file
- **certificates** (*list[str]*) – The PEM-encoded certificates, or a list of paths of the PEM-encoded certificate files

```
admin.ssl.import_from_chain(
    r'C:\Users\jsmith\Downloads\private.key',
    r'C:\Users\jsmith\Downloads\domain.crt',
    r'C:\Users\jsmith\Downloads\intermediate.crt',
    r'C:\Users\jsmith\Downloads\root.crt'
)
```

### 2.1.1.19 Logs

`Logs.get(topic='system', min_severity='info', origin_type='Portal', origin=None, before=None, after=None, filters=None)`

Get logs from the Portal

#### Parameters

- **topic** (`cterasdk.core.enum.LogTopic`, *optional*) – Log topic to get, defaults to `cterasdk.core.enum.LogTopic.System`
- **min\_severity** (`cterasdk.core.enum.Severity`, *optional*) – Minimum severity of logs to get, defaults to `cterasdk.core.enum.Severity.INFO`
- **origin\_type** (`cterasdk.core.enum.OriginType`, *optional*) – Origin type of the logs to get, defaults to `cterasdk.core.enum.OriginType.Portal`
- **origin** (*str*, *optional*) – Log origin (e.g. device name, Portal server name), defaults to `None`
- **before** (*str*, *optional*) – Get logs before this date (in format “%m/%d/%Y %H:%M:%S”), defaults to `None`
- **after** (*str*, *optional*) – Get logs after this date (in format “%m/%d/%Y %H:%M:%S”), defaults to `None`
- **filters** (*list*[`cterasdk.core.query.FilterBuilder`], *optional*) – List of additional filters, defaults to `None`

**Returns** Iterator for all matching logs

**Return type** `cterasdk.lib.iterator.Iterator`[`cterasdk.object.Object`]

`Logs.device(name, topic='system', min_severity='info', before=None, after=None, filters=None)`

Get device logs from the Portal

#### Parameters

- **name** (*str*) – Name of a device
- **topic** (`cterasdk.core.enum.LogTopic`, *optional*) – Log topic to get, defaults to `cterasdk.core.enum.LogTopic.System`
- **min\_severity** (`cterasdk.core.enum.Severity`, *optional*) – Minimum severity of logs to get, defaults to `cterasdk.core.enum.Severity.INFO`
- **before** (*str*, *optional*) – Get logs before this date (in format “%m/%d/%Y %H:%M:%S”), defaults to `None`
- **after** (*str*, *optional*) – Get logs after this date (in format “%m/%d/%Y %H:%M:%S”), defaults to `None`
- **filters** (*list*[`cterasdk.core.query.FilterBuilder`], *optional*) – List of additional filters, defaults to `None`

**Returns** Iterator for all matching logs

**Return type** `cterasdk.lib.iterator.Iterator[cterasdk.object.Object]`

```
"""Retrieve all cloud backup logs for device 'WIN-SRV2019'"""
admin.logs.device('WIN-SRV2019', topic='backup')
```

### 2.1.1.20 Syslog

`Syslog.is_enabled()`

Check if forwarding log messages over syslog is enabled

`Syslog.get_configuration()`

Retrieve the syslog server configuration

`Syslog.enable(server, port=514, protocol='UDP', min_severity='info')`

Enable Syslog

#### Parameters

- **server** (*str*) – Syslog server address
- **port** (*int, optional*) – Syslog server port
- **protocol** (`cterasdk.core.enum.IPProtocol`, *optional*) – Syslog server IP protocol
- **min\_severity** (`cterasdk.core.enum.Severity`, *optional*) – Minimum log severity to forward

`Syslog.disable()`

## 2.1.2 End User Portal

### Table of Contents

- *End User Portal*
  - *Instantiate a Services Portal object*
  - \* *Logging in*

### 2.1.2.1 Instantiate a Services Portal object

`class cterasdk.object.Portal.ServicesPortal(host, port=None, https=True)`

Main class for Service operations on a Portal

`__init__(host, port=None, https=True)`

#### Parameters

- **host** (*str*) – The fully qualified domain name, hostname or an IPv4 address of the Portal
- **port** (*int, optional*) – Set a custom port number (0 - 65535), If not set defaults to 80 for http and 443 for https
- **https** (*bool, optional*) – Set to True to require HTTPS, defaults to True

```
user = ServicesPortal('portal.ctera.com') # will use HTTPS over port 443
```

**Warning:** for any certificate related error, this library will prompt for your consent in order to proceed. to avoid the prompt, you may configure *chopin-core* to automatically trust the server's certificate, using: `config.http['ssl'] = 'Trust'`

### Logging in

`ServicesPortal.test()`

Verification check to ensure the target host is a Portal.

```
user.test()
```

`ServicesPortal.login(username, password)`

Log in

#### Parameters

- **username** (*str*) – User name to log in
- **password** (*str*) – User password

```
user.login('walice', 'G3neralZ0d!')
```

`ServicesPortal.logout()`

Log out

```
user.logout()
```

### 2.1.3 File Browser

#### Table of Contents

- *File Browser*
  - *File Browser*
    - \* *List*
    - \* *Download*
    - \* *Copy*
    - \* *Create Public Link*
  - *Cloud Drive*
    - \* *Create Directory*
    - \* *Rename*
    - \* *Delete*
    - \* *Undelete*



- \* *Move*
- \* *Upload*
- \* *Collaboration Shares*
- *Backups*

### 2.1.3.1 File Browser

#### List

`FileBrowser.ls(path, include_deleted=False)`

Execute ls on the provided path

##### Parameters

- **path** (*str*) – Path to list
- **include\_deleted** (*bool, optional*) – Include deleted files, defaults to False

```
file_browser.ls('') # List the contents of the Cloud Drive
file_browser.ls('My Files') # List the contents of the 'My Files' folder
file_browser.ls('My Files', True) # Include deleted files
```

`FileBrowser.walk(path, include_deleted=False)`

Perform walk on the provided path

##### Parameters

- **path** (*str*) – Path to perform walk on
- **include\_deleted** (*bool, optional*) – Include deleted files, defaults to False

```
file_browser.walk('My Files')
```

#### Download

`FileBrowser.download(path, destination=None)`

Download a file

##### Parameters

- **path** (*str*) – Path of the file to download
- **destination** (*str, optional*) – File destination, if it is a directory, the original filename will be kept, defaults to the default directory

```
file_browser.download('My Files/Documents/Sample.docx')
```

### Copy

`FileBrowser.copy(src, dest)`

Copy a file or directory

#### Parameters

- **src** (*str*) – The source path of the file or directory
- **dst** (*str*) – The destination path of the file or directory

```
file_browser.copy('My Files/Documents/Sample.docx', 'The/quick/brown/fox')
```

`FileBrowser.copy_multi(src, dest)`

```
file_browser.copy_multi(['My Files/Documents/Sample.docx', 'My Files/Documents/Burndown.↵xlsx'], 'The/quick/brown/fox')
```

### Create Public Link

`FileBrowser.mklink(path, access='RO', expire_in=30)`

Create a link to a file

#### Parameters

- **path** (*str*) – The path of the file to create a link to
- **access** (*str, optional*) – Access policy of the link, defaults to 'RO'
- **expire\_in** (*int, optional*) – Number of days until the link expires, defaults to 30

```
"""
Access:
- RW: Read Write
- RO: Read Only
- NA: No Access
"""

"""Create a Read Only public link to a file that expires in 30 days"""
file_browser.mklink('My Files/Documents/Sample.docx')

"""Create a Read Write public link to a folder that expires in 45 days"""
file_browser.mklink('My Files/Documents/Sample.docx', 'RW', 45)
```

**Warning:** you cannot use this tool to create read write public links to files.

### 2.1.3.2 Cloud Drive

The CloudDrive class is a subclass to `cterasdk.common.files.browser.FileBrowser` providing file access to the user's Cloud Drive

```
from getpass import getpass

"""Accessing Cloud Drive Files and Folders as a Global Administrator"""
admin = GlobalAdmin('portal.ctera.com') # logging in to /admin
admin.login('admin', getpass())
file_browser = admin.files # the field is an instance of CloudDrive class object

"""Accessing Cloud Drive Files and Folders as a Tenant User Account"""
user = ServicesPortal('portal.ctera.com') # logging in to /ServicesPortal
user.login('bwayne', getpass())
file_browser = user.files # the field is an instance of CloudDrive class object
```

#### Create Directory

CloudDrive.**mkdir**(*path*, *recurse=False*)  
Create a new directory

##### Parameters

- **path** (*str*) – Path of the directory to create
- **recurse** (*bool, optional*) – Whether to create the path recursively, defaults to False

```
file_browser.mkdir('My Files/Documents')

file_browser.mkdir('The/quick/brown/fox', recurse = True)
```

#### Rename

CloudDrive.**rename**(*path*, *name*)  
Rename a file

##### Parameters

- **path** (*str*) – Path of the file or directory to rename
- **name** (*str*) – The name to rename to

```
file_browser.rename('My Files/Documents/Sample.docx', 'Wizard Of Oz.docx')
```

### Delete

CloudDrive.**delete**(*path*)

Delete a file

**Parameters** *path* (*str*) – Path of the file or directory to delete

```
file_browser.delete('My Files/Documents/Sample.docx')
```

CloudDrive.**delete\_multi**(\**args*)

Delete multiple files and/or directories

**Parameters** \**args* – Variable lengthed list of paths of files and/or directories to delete

```
file_browser.delete_multi('My Files/Documents/Sample.docx', 'The/quick/brown/fox')
```

### Undelete

CloudDrive.**undelete**(*path*)

Restore a previously deleted file or directory

**Parameters** *path* (*str*) – Path of the file or directory to restore

```
file_browser.undelete('My Files/Documents/Sample.docx')
```

CloudDrive.**undelete\_multi**(\**args*)

Restore previously deleted multiple files and/or directories

**Parameters** \**args* – Variable length list of paths of files and/or directories to restore

```
file_browser.undelete_multi('My Files/Documents/Sample.docx', 'The/quick/brown/fox')
```

### Move

CloudDrive.**move**(*src*, *dest*)

Move a file or directory

**Parameters**

- **src** (*str*) – The source path of the file or directory
- **dst** (*str*) – The destination path of the file or directory

```
file_browser.move('My Files/Documents/Sample.docx', 'The/quick/brown/fox')
```

CloudDrive.**move\_multi**(*src*, *dest*)

```
file_browser.move_multi(['My Files/Documents/Sample.docx', 'My Files/Documents/Burndown.  
↪xlsx'], 'The/quick/brown/fox')
```

## Upload

`CloudDrive.upload(file_path, server_path)`

Upload a file

### Parameters

- **file\_path** (*str*) – Path to the local file to upload
- **server\_path** (*str*) – Path to the directory to upload the file to

```

"""
Upload the 'Tree.jpg' file as an End User to 'Forest' directory
"""
file_browser.files.upload(r'C:\Users\BruceWayne\Downloads\Tree.jpg', 'Images/Forest')

"""
Upload the 'Tree.jpg' file as an Administrator to an End User's Cloud Drive
"""
file_browser.files.upload(r'C:\Users\Administrator\Downloads\Tree.jpg', 'Bruce Wayne/
↪Images/Forest')

```

## Collaboration Shares

`CloudDrive.share(path, recipients, as_project=True, allow_reshare=True, allow_sync=True)`

Share a file or a folder

### Parameters

- **path** (*str*) – The path of the file or folder to share
- **recipients** (*list[cterasdk.core.types.ShareRecipient]*) – A list of share recipients
- **as\_project** (*bool, optional*) – Share as a team project, defaults to True when the item is a cloud folder else False
- **allow\_reshare** (*bool, optional*) – Allow recipients to re-share this item, defaults to True
- **allow\_sync** (*bool, optional*) – Allow recipients to sync this item, defaults to True when the item is a cloud folder else False

**Returns** A list of all recipients added to the collaboration share

**Return type** `list[cterasdk.core.types.ShareRecipient]`

```

"""
Share with a local user and a local group.
- Grant the local user with read only access for 30 days
- Grant the local group with read write access with no expiration
"""

alice = portal_types.UserAccount('alice')
engineers = portal_types.GroupAccount('Engineers')

recipients = []

```

(continues on next page)

(continued from previous page)

```
alice_rcpt = portal_types.ShareRecipient.local_user(alice).expire_in(30).read_only()
engineers_rcpt = portal_types.ShareRecipient.local_group(engineering).read_write()

file_browser.share('Codebase', [alice_rcpt, engineers_rcpt])
```

```
"""
Share with an external recipient
- Grant the external user with preview only access for 10 days
"""
jsmith = portal_types.ShareRecipient.external('jsmith@hotmail.com').expire_in(10).
↳preview_only()
file_browser.share('My Files/Projects/2020/ProjectX', [jsmith])

"""
Share with an external recipient, and require 2 factor authentication
- Grant the external user with read only access for 5 days, and require 2 factor_
↳authentication over e-mail
"""
jsmith = portal_types.ShareRecipient.external('jsmith@hotmail.com', True).expire_in(5).
↳read_only()
file_browser.share('My Files/Projects/2020/ProjectX', [jsmith])
```

```
"""
Share with a domain groups
- Grant the Albany domain group with read write access with no expiration
- Grant the Cleveland domain group with read only access with no expiration
"""
albany_group = portal_types.GroupAccount('Albany', 'ctera.com')
cleveland_group = portal_types.GroupAccount('Cleveland', 'ctera.com')

albany_rcpt = portal_types.ShareRecipient.domain_group(albany_group).read_write()
cleveland_rcpt = portal_types.ShareRecipient.domain_group(cleveland_group).read_only()

file_browser.share('Cloud/Albany', [albany_rcpt, cleveland_rcpt])
```

`CloudDrive.add_share_recipients(path, recipients)`

Add share recipients

#### Parameters

- **path** (*str*) – The path of the file or folder
- **recipients** (*list[cterasdk.core.types.ShareRecipient]*) – A list of share recipients

**Returns** A list of all recipients added

**Return type** `list[cterasdk.core.types.ShareRecipient]`

---

**Note:** if the share recipients provided as an argument already exist, they will be skipped and not updated

---

`CloudDrive.remove_share_recipients(path, accounts)`

Remove share recipients

**Parameters**

- **path** (*str*) – The path of the file or folder
- **accounts** (*list[cterasdk.core.types.PortalAccount]*) – A list of portal user or group accounts

**Returns** A list of all share recipients removed

**Return type** *list[cterasdk.core.types.PortalAccount]*

CloudDrive.**unshare**(*path*)

Unshare a file or a folder

```
"""
Unshare a file or a folder
"""
file_browser.unshare('Codebase')
file_browser.unshare('My Files/Projects/2020/ProjectX')
file_browser.unshare('Cloud/Albany')
```

### 2.1.3.3 Backups

The Backups class is a subclass to `cterasdk.common.files.browser.FileBrowser` providing access to files stored in backup folders

```
from getpass import getpass

"""Accessing Backups as a Global Administrator"""
admin = GlobalAdmin('portal.ctera.com') # logging in to /admin
admin.login('admin', getpass())
file_browser = admin.files # the field is an instance of Backups class object

"""Accessing Backups as a Tenant User Account"""
user = ServicesPortal('portal.ctera.com') # logging in to /ServicesPortal
user.login('bwayne', getpass())
file_browser = user.backups # the field is an instance of Backups class object
```

## 2.2 Edge Filer

### 2.2.1 Gateway

**Table of Contents**

- *Gateway*
  - *Instantiate a Gateway object*
    - \* *Logging in*
    - \* *Core Methods*
    - \* *Device Configuration*

- \* *Code Snippets*
- \* *Storage*
  - *Format*
  - *Volumes*
- \* *Shares*
- \* *Users*
- \* *Groups*
- \* *Active Directory*
- \* *Cloud Services*
- \* *Applying a License*
- \* *Caching*
- \* *Cloud Backup*
  - *Cloud Backup Files*
- \* *Cloud Sync*
  - *Cloud Sync Bandwidth Throttling*
- \* *File Access Protocols*
  - *Windows File Sharing (CIFS/SMB)*
- \* *Network*
  - *Network Diagnostics*
- \* *Mail Server*
- \* *Logging*
  - *SMB Audit Logs*
- \* *Reset*
- \* *SSL*
- \* *Power Management*
- \* *SNMP*
- \* *Support*
  - *Support Report*
  - *Debug*
  - *Telnet Access*
  - *SSH Access*



### 2.2.1.1 Instantiate a Gateway object

**class** `cterasdk.object.Gateway.Gateway`(*host, port=None, https=False, Portal=None*)

Main class operating on a Gateway

#### Variables

- **config** (`cterasdk.edge.config.Config`) – Object holding the Gateway Configuration APIs
- **network** (`cterasdk.edge.network.Network`) – Object holding the Gateway Network APIs
- **licenses** (`cterasdk.edge.licenses.Licenses`) – Object holding the Gateway Licenses APIs
- **services** (`cterasdk.edge.services.Services`) – Object holding the Gateway Services APIs
- **directoryservice** (`cterasdk.edge.directoryservice.DirectoryService`) – Object holding the Gateway Active Directory APIs
- **telnet** (`cterasdk.edge.telnet.Telnet`) – Object holding the Gateway Telnet APIs
- **syslog** (`cterasdk.edge.syslog.Syslog`) – Object holding the Gateway Syslog APIs
- **tasks** (`cterasdk.edge.taskmgr.Tasks`) – Object holding the Gateway Background Tasks APIs
- **audit** (`cterasdk.edge.audit.Audit`) – Object holding the Gateway Audit APIs
- **mail** (`cterasdk.edge.mail.Mail`) – Object holding the Gateway Mail APIs
- **backup** (`cterasdk.edge.backup.Backup`) – Object holding the Gateway Backup APIs
- **sync** (`cterasdk.edge.sync.Sync`) – Object holding the Gateway Sync APIs
- **cache** (`cterasdk.edge.cache.Cache`) – Object holding the Gateway Cache APIs
- **snmp** (`cterasdk.edge.snmp.SNMP`) – Object holding the Gateway SNMP APIs
- **ssl** (`cterasdk.edge.ssl.SSL`) – Object holding the Gateway SSL APIs
- **ssh** (`cterasdk.edge.ssh.SSH`) – Object holding the Gateway SSH APIs
- **power** (`cterasdk.edge.power.Power`) – Object holding the Gateway Power APIs
- **users** (`cterasdk.edge.users.Users`) – Object holding the Gateway Users APIs
- **groups** (`cterasdk.edge.groups.Groups`) – Object holding the Gateway Groups APIs
- **drive** (`cterasdk.edge.drive.Drive`) – Object holding the Gateway Drive APIs
- **volumes** (`cterasdk.edge.volumes.Volumes`) – Object holding the Gateway Volumes APIs
- **array** (`cterasdk.edge.array.Array`) – Object holding the Gateway Array APIs
- **shares** (`cterasdk.edge.shares.Shares`) – Object holding the Gateway Shares APIs
- **smb** (`cterasdk.edge.smb.SMB`) – Object holding the Gateway SMB APIs
- **aio** (`cterasdk.edge.aio.AIO`) – Object holding the Gateway AIO APIs
- **ftp** (`cterasdk.edge.ftp.FTP`) – Object holding the Gateway FTP APIs
- **afp** (`cterasdk.edge.afp.AFP`) – Object holding the Gateway AFP APIs

- **nfs** (`cterasdk.edge.nfs.NFS`) – Object holding the Gateway NFS APIs
- **rsync** (`cterasdk.edge.rsync.RSync`) – Object holding the Gateway RSync APIs
- **timezone** (`cterasdk.edge.timezone.Timezone`) – Object holding the Gateway Timezone APIs
- **logs** (`cterasdk.edge.logs.Logs`) – Object holding the Gateway Logs APIs
- **ntp** (`cterasdk.edge.ntp.NTP`) – Object holding the Gateway NTP APIs
- **shell** (`cterasdk.edge.shell.Shell`) – Object holding the Gateway Shell APIs
- **cli** (`cterasdk.edge.cli.CLI`) – Object holding the Gateway CLI APIs
- **support** (`cterasdk.edge.support.Support`) – Object holding the Gateway Support APIs
- **files** (`cterasdk.edge.files.FileBrowser`) – Object holding the Gateway File Browsing APIs
- **firmware** (`cterasdk.edge.firmware.Firmware`) – Object holding the Gateway Firmware APIs

`__init__(host, port=None, https=False, Portal=None)`

#### Parameters

- **host** (*str*) – The fully qualified domain name, hostname or an IPv4 address of the Gateway
- **port** (*int, optional*) – Set a custom port number (0 - 65535), If not set defaults to 80 for http and 443 for https
- **https** (*bool, optional*) – Set to True to require HTTPS, defaults to False
- **Portal** (`cterasdk.object.Portal.Portal`, *optional*) – The portal through which the remote session was created, defaults to None

```
filer = Gateway('10.100.102.4') # will use HTTP over port 80
filer = Gateway('10.100.102.4', 8080) # will use HTTP over port 8080
filer = Gateway('vGateway-0dbc', 443, True) # will use HTTPS over port 443
```

**Warning:** for any certificate related error, this library will prompt for your consent in order to proceed. to avoid the prompt, you may configure *chopin-core* to automatically trust the server's certificate, using: `config.http['ssl'] = 'Trust'`

## Logging in

`Gateway.test()`

Verification check to ensure the target host is a Gateway.

```
filer.test()
```

`Gateway.login(username, password)`

Log in

#### Parameters

- **username** (*str*) – User name to log in
- **password** (*str*) – User password

```
filer.login('admin', 'G3neralZ0d!')
```

Gateway.**logout**()  
Log out

```
filer.logout()
```

Gateway.**whoami**()  
Return the name of the logged in user.

**Return cterasdk.common.object.Object** The session object of the current user

```
filer.whoami()
```

## Core Methods

Gateway.**show**(*path, use\_file\_url=False*)  
Print a schema object as a JSON string.

```
filer.show('/status/storage/volumes')
```

Gateway.**show\_multi**(*path, paths, use\_file\_url=False*)  
Print one or more schema objects as a JSON string.

```
filer.show_multi(['/config/storage/volumes', '/status/storage/volumes'])
```

Gateway.**get**(*path, params=None, use\_file\_url=False*)  
Retrieve a schema object as a Python object.

```
"""Retrieve the device configuration and print it as JSON string"""
config = filer.get('/config')
print(config)

"""Retrieve the device settings and print the hostname and location settings"""
settings = filer.get('/config/device')

print(settings.hostname)
print(settings.location)

"""Retrieve a list of volumes and print the name of the first volume"""
volumes = filer.get('/status/storage/volumes') # returns a list of volumes
print(volumes[0].name) # will print the name of the first volume

"""Retrieve the network settings and print the MTU setting"""
network = filer.get('/config/network') # returns network settings
```

(continues on next page)

(continued from previous page)

```
print(network.ports[0].ethernet.mtu) # will print the MTU setting
```

Gateway.**get\_multi**(*path, paths, use\_file\_url=False*)  
Retrieve one or more schema objects as a Python object.

```
"""Retrieve '/config/cloudsync' and '/proc/cloudsync' at once"""
device = filer.get_multi(['/config/cloudsync', '/proc/cloudsync'])
print(device.config.cloudsync.cloudExtender.operationMode)
print(device.proc.cloudsync.serviceStatus.uploadingFiles)
```

Gateway.**put**(*path, value, use\_file\_url=False*)  
Update a schema object or attribute.

```
"""Disable the first time wizard"""
filer.put('/config/gui/openFirstTimeWizard', False)
"""Turn off FTP access on all shares"""
shares = filer.get('/config/fileservices/share')
for share in shares:
    share.exportToFTP = False
    filer.put('/config/fileservices/share/' + share.name, share)
```

Gateway.**execute**(*path, name, param=None, use\_file\_url=False*)  
Execute a schema object method.

```
"""Execute the file-eviction process"""
filer.execute('/config/cloudsync', 'forceExecuteEvictor') # doesn't require a param
"""Reboot the Gateway"""
filer.execute('/statuc/device', 'reboot') # doesn't require a param
"""TCP Connect"""
param = Object()
param.address = 'chopin.ctera.com'
param.port = 995 # CTP
bgTask = filer.execute('/status/network', 'tcpconnect', param)
print(bgTask)
```

**See also:**

Execute the file-eviction process: `cterasdk.edge.cache.Cache.force_eviction()`, Reboot the Gateway: `cterasdk.edge.power.reboot()`, Execute tcp connect: `Gateway.tcp_connect()`

`Gateway.add(path, param, use_file_url=False)`  
Add a schema object.

```
"""Add a user account"""
user = Object()
user.username = 'mickey'
user.fullName = 'Mickey Mouse'
user.email = 'm.mouse@disney.com'
user.uid = 1940
user.password = 'M!niM0us3'
filer.add('/config/auth/users', user)
```

`Gateway.delete(path, use_file_url=False)`  
Delete a schema object.

```
"""Delete a user account"""
user = 'mickey'
filer.delete('/config/auth/users/' + user)
```

**Device Configuration**

`Config.get_hostname()`  
Get the hostname of the gateway

**Return str** The hostname of the gateway

```
hostname = filer.config.hostname()
```

`Config.set_hostname(hostname)`  
Set the hostname of the gateway

**Parameters hostname (str)** – New hostname to set

**Return str** The new hostname

```
filer.config.set_hostname('Chopin')
```

`Config.get_location()`  
Get the location of the gateway

**Return str** The location of the gateway

```
location = filer.config.location()
```

`Config.set_location(location)`

Set the location of the gateway

**Parameters** `location` (*str*) – New location to set

**Return** `str` The new location

```
filer.config.set_location('Jupiter')
```

`Config.disable_wizard()`

Disable the first time wizard

```
filer.config.disable_wizard()
```

`Config.export(destination=None)`

Export the Edge Filer configuration

**Parameters** `destination` (*str, optional*) – File destination, defaults to the default directory

```
filer.config.export()
```

`Config.import_config(config, exclude=None)`

Import the Edge Filer configuration

**Parameters**

- **config** (*str*) – A string or a path to the Edge Filer configuration file
- **delete\_attrs** (*list[str], optional*) – List of configuration properties to exclude from import

```
"""Import Edge Filer configuration from file"""
filer.config.import_config(r'C:\Users\bwayne\Downloads\EdgeFiler.xml')

"""Import configuration without network settings"""
filer.config.import_config(r'C:\Users\bwayne\Downloads\EdgeFiler.xml', exclude=[
    '/config/network'
])

"""Import configuration without the 'logs' and 'public' shares"""
filer.config.import_config(r'C:\Users\bwayne\Downloads\EdgeFiler.xml', exclude=[
    '/config/fileservices/share/logs',
    '/config/fileservices/share/public'
])
```

## Code Snippets

```
filer.get('/status/device/runningFirmware') # Get firmware version

"""Get the entire device status object"""
status = filer.get('/status/device')
print(status.runningFirmware, status.MacAddress)
```

## Storage

### Format

Drive.**format**(*name*)  
Format a drive

**Parameters** *name* (*str*) – The name of the drive to format

```
filer.drive.format('SATA1')
```

Drive.**format\_all**()  
Format all drives

```
filer.drive.format_all()
```

## Volumes

Volumes.**add**(*name*, *size=None*, *filesystem='xfs'*, *device=None*, *passphrase=None*)  
Add a new volume to the gateway

### Parameters

- **name** (*str*) – Name of the new volume
- **size** (*int*, *optional*) – Size of the new volume, defaults to the device's size
- **filesystem** (*str*, *optional*) – Filesystem to use, defaults to xfs
- **device** (*str*, *optional*) – Name of the device to use for the new volume, can be left as None if there the gateway has only one
- **passphrase** (*str*, *optional*) – Passphrase for the volume

**Returns** Gateway response

```
filer.volumes.add('localcache')
```

Volumes.**delete**(*name*)  
Delete a volume

**Parameters** *name* (*str*) – Name of the volume to delete

```
filer.volumes.delete('localcache')
```

Volumes.**delete\_all**()  
Delete all volumes

```
filer.volumes.delete_all()
```

## Shares

```
Shares.add(name, directory, acl=None, access='winAclMode', csc='manual', dir_permissions=777,
           comment=None, export_to_afp=False, export_to_ftp=False, export_to_nfs=False,
           export_to_pc_agent=False, export_to_rsync=False, indexed=False, trusted_nfs_clients=None,
           uuid=None)
```

Add a network share.

### Parameters

- **name** (*str*) – The share name
- **directory** (*str*) – Full directory path
- **acl** (*list*[`cterasdk.edge.types.ShareAccessControlEntry`]) – List of access control entries
- **access** (`cterasdk.edge.enum.Acl`) – The Windows File Sharing authentication mode, defaults to `winAclMode`
- **csc** (`cterasdk.edge.enum.ClientSideCaching`) – The client side caching (offline files) configuration, defaults to `manual`
- **dir\_permissions** (*int*) – Directory Permission, defaults to `777`
- **comment** (*str*) – Comment
- **export\_to\_afp** (*bool*) – Whether to enable AFP access, defaults to `False`
- **export\_to\_ftp** (*bool*) – Whether to enable FTP access, defaults to `False`
- **export\_to\_nfs** (*bool*) – Whether to enable NFS access, defaults to `False`
- **export\_to\_pc\_agent** (*bool*) – Whether to allow as a destination share for CTERA Backup Agents, defaults to `False`
- **export\_to\_rsync** (*bool*) – Whether to enable access over rsync, defaults to `False`
- **indexed** (*bool*) – Whether to enable indexing for search, defaults to `False`
- **trusted\_nfs\_clients** (*list*[`cterasdk.edge.types.NFSv3AccessControlEntry`]) – Trusted NFS v3 clients, defaults to `None`

```
"""
Create an ACL-enabled cloud share called 'Accounting' and define four access control
↪ entries:
```

- 1) Everyone - Read Only (Local Group)
- 2) admin - Read Write (Local User)
- 3) Domain Admins - Read Only (Domain Group)
- 4) bruce.wayne@ctera.com - Read Write (Domain User)

Principal Type:

- LG: Local Group
- LU: Local User
- DG: Domain Group
- DU: Domain User

(continues on next page)



(continued from previous page)

```

Access:
- RW: Read Write
- RO: Read Only
- NA: No Access
"""

everyone = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.LG, 'Everyone
↳ ', gateway_enum.FileAccessMode.RO)
local_admin = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.LU, 'admin
↳ ', gateway_enum.FileAccessMode.RW)
domain_admins = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DG,
↳ 'CTERA\Domain Admins', gateway_enum.FileAccessMode.RO)
bruce_wayne = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DU,
↳ 'bruce.wayne@ctera.com', gateway_enum.FileAccessMode.RW)

filer.shares.add('Accounting', 'cloud/users/Service Account/Accounting', acl = [ \
    everyone, local_admin, domain_admins, bruce_wayne \
])

"""Create an 'Only Authenticated Users' cloud share called 'FTP' and enable FTP access to
↳ everyone"""

everyone = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.LG, 'Everyone
↳ ', gateway_enum.FileAccessMode.RW)

filer.shares.add('FTP', 'cloud/users/Service Account/FTP', acl = [everyone], export_to_
↳ ftp = True)

"""Add an NFS share and enable access to two hosts"""
nfs_client_1 = gateway_types.NFSv3AccessControlEntry('192.168.0.1', '255.255.255.0',
↳ gateway_enum.FileAccessMode.RW) # read write
nfs_client_2 = gateway_types.NFSv3AccessControlEntry('192.168.0.2', '255.255.255.0',
↳ gateway_enum.FileAccessMode.RO) # read only
filer.shares.add('NFS', 'cloud/users/Service Account/NFS', export_to_nfs=True, trusted_
↳ nfs_clients=[nfs_client_1, nfs_client_2])

```

**Shares.add\_acl**(name, acl)

Add one or more access control entries to an existing share.

#### Parameters

- **name** (str) – The share name
- **acl** (list[cterasdk.edge.types.ShareAccessControlEntry]) – List of access control entries to add

```

"""Add two access control entries to the 'Accounting' share"""

domain_group = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DG,
↳ 'CTERA\leadership', gateway_enum.FileAccessMode.RW)
domain_user = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DU,
↳ 'clark.kent@ctera.com', gateway_enum.FileAccessMode.RO)

```

(continues on next page)

(continued from previous page)

```
filer.shares.add_acl('Accounting', [domain_group, domain_user])
```

`Shares.set_acl(name, acl)`

Set a network share's access control entries.

#### Parameters

- **name** (*str*) – The share name
- **acl** (*list*[`cterasdk.edge.types.ShareAccessControlEntry`]) – List of access control entries

**Warning:** this method will override the existing access control entries

```
"""Set the access control entries of the 'Accounting' share"""
domain_group = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DG,
↳ 'CTERA\leadership', gateway_enum.FileAccessMode.RW)
domain_user = gateway_types.ShareAccessControlEntry(gateway_enum.PrincipalType.DU,
↳ 'clark.kent@ctera.com', gateway_enum.FileAccessMode.RO)
filer.shares.set_acl('Accounting', [domain_group, domain_user])
```

`Shares.remove_acl(name, acl)`

Remove one or more access control entries from an existing share.

#### Parameters

- **name** (*str*) – The share name
- **acl** (*list*[`cterasdk.edge.types.RemoveShareAccessControlEntry`]) – List of access control entries to remove

```
"""Remove access control entries from the 'Accounting' share"""
domain_group = gateway_types.RemoveShareAccessControlEntry(gateway_enum.PrincipalType.DG,
↳ 'CTERA\leadership')
domain_user = gateway_types.RemoveShareAccessControlEntry(gateway_enum.PrincipalType.DU,
↳ 'clark.kent@ctera.com')
filer.shares.remove_acl('Accounting', [domain_group, domain_user])
```

`Shares.set_share_winacls(name)`

Set a network share to use Windows ACL Emulation Mode

**Parameters** **name** (*str*) – The share name

```
filer.shares.set_share_winacls('cloud')
```

`Shares.block_files(name, extensions)`

Configure a share to block one or more file extensions

#### Parameters

- **name** (*str*) – The share name

- **extensions** (*list[str]*) – List of file extensions to block

```
filer.shares.block_files('Accounting', ['exe', 'cmd', 'bat'])
```

**Shares.modify**(*name, directory=None, acl=None, access=None, csc=None, dir\_permissions=None, comment=None, export\_to\_afp=None, export\_to\_ftp=None, export\_to\_nfs=None, export\_to\_pc\_agent=None, export\_to\_rsync=None, indexed=None, trusted\_nfs\_clients=None*)

Modify an existing network share. All parameters but name are optional and default to None

#### Parameters

- **name** (*str*) – The share name
- **directory** (*str, optional*) – Full directory path
- **acl** (*list[cterasdk.edge.types.ShareAccessControlEntry], optional*) – List of access control entries
- **access** (*cterasdk.edge.enum.Acl, optional*) – The Windows File Sharing authentication mode
- **csc** (*cterasdk.edge.enum.ClientSideCaching, optional*) – The client side caching (offline files) configuration
- **dir\_permissions** (*int, optional*) – Directory Permission
- **comment** (*str, optional*) – Comment
- **export\_to\_afp** (*bool, optional*) – Whether to enable AFP access
- **export\_to\_ftp** (*bool, optional*) – Whether to enable FTP access
- **export\_to\_nfs** (*bool, optional*) – Whether to enable NFS access
- **export\_to\_pc\_agent** (*bool, optional*) – Whether to allow as a destination share for CTERA Backup Agents
- **export\_to\_rsync** (*bool, optional*) – Whether to enable access over rsync
- **indexed** (*bool, optional*) – Whether to enable indexing for search
- **trusted\_nfs\_clients** (*list[cterasdk.edge.types.NFSv3AccessControlEntry]*) – Trusted NFS v3 clients, defaults to None

```
""" Disable all file-access protocols on all shares """
shares = filer.shares.get() # obtain a list of all shares

for share in shares:
    filer.share.modify(
        share.name,
        export_to_afp=False,      # Apple File Sharing
        export_to_ftp=False,     # FTP
        export_to_nfs=False,     # NFS
        export_to_pc_agent=False, # CTERA Agent
        export_to_rsync=False,   # rsync
        indexed=False           # Search
    )
```

**Shares.delete**(*name*)

Delete a share.

**Parameters** **name** (*str*) – The share name

```
filer.shares.delete('Accounting')
```

## Users

`Users.add(username, password, full_name=None, email=None, uid=None)`  
Add a user of the Gateway

### Parameters

- **username** (*str*) – User name for the new user
- **password** (*str*) – Password for the new user
- **full\_name** (*str, optional*) – The full name of the new user, defaults to None
- **email** (*str, optional*) – E-mail address of the new user, defaults to None
- **uid** (*str, optional*) – The uid of the new user, defaults to None

```
filer.users.add('Clark', 'Kryptonite1!') # without a full name, email or custom uid
filer.users.add('alice', 'W!z4rd0f0z!', 'Alice Wonderland') # including a full name
filer.users.add('Bruce', 'GothamCity1!', 'Bruce Wayne', 'bruce.wayne@we.com', uid = ↵
↵1940) # all
```

`Users.modify(username, password=None, full_name=None, email=None, uid=None)`  
Modify an existing user of the Gateway

### Parameters

- **username** (*str*) – User name to modify
- **password** (*str, optional*) – New password, defaults to None
- **full\_name** (*str, optional*) – The full name of the user, defaults to None
- **email** (*str, optional*) – E-mail address of the user, defaults to None
- **uid** (*str, optional*) – The uid of the user, defaults to None

```
filer.users.modify('Clark', 'Passw0rd1!') # Change a user's password
filer.users.modify('Clark', email='clark.kent@krypton.com') # Change a user's email
```

`Users.delete(username)`  
Delete an existing user

**Parameters** **username** (*str*) – User name of the user to delete

```
filer.users.delete('alice')
```

`Users.add_first_user(username, password, email="")`  
Add the first user of the Gateway and login

### Parameters

- **username** (*str*) – User name for the new user
- **password** (*str*) – Password for the new user
- **email** (*str, optional*) – E-mail address of the new user, defaults to an empty string

```
filer.users.add_first_user('admin', 'L3tsG3tR34dyT0Rumb13!')
```

## Groups

`Groups.add_members(group, members)`  
Add members to a group

### Parameters

- **group** (*str*) – Name of the group
- **members** (*list[cterasdk.edge.types.UserGroupEntry]*) – List of users and groups to add to the group

```
"""Add Bruce Wayne to the local Administrators group"""
member = gateway_types.UserGroupEntry(gateway_enum.PrincipalType.DU, 'bruce.wayne@we.com
↪')
filer.groups.add_members('Administrators', [member])

"""Add Bruce Wayne and Domain Admins to the local Administrators group"""

domain_user = gateway_types.UserGroupEntry(gateway_enum.PrincipalType.DU, 'bruce.
↪wayne@we.com')
domain_group = gateway_types.UserGroupEntry(gateway_enum.PrincipalType.DG, 'WE\Domain
↪Admins')
filer.groups.add_members('Administrators', [domain_user, domain_group])
```

`Groups.remove_members(group, members)`  
Remove members from a group

### Parameters

- **group** (*str*) – Name of the group
- **members** (*list[cterasdk.edge.types.UserGroupEntry]*) – List of users and groups to remove from the group

```
"""Remove Bruce Wayne from the local Administrators group"""

filer.groups.remove_members('Administrators', [('DU', 'bruce.wayne@we.com')])

"""Remove Bruce Wayne and Domain Admins from the local Administrators group"""

filer.groups.remove_members('Administrators', [('DU', 'bruce.wayne@we.com'), ('DG', 'WE\
↪Domain Admins')])
```

### Active Directory

`DirectoryService.connect(domain, username, password, ou=None, check_connection=False)`

Connect the Gateway to an Active Directory

#### Parameters

- **domain** (*str*) – The active directory domain to connect to
- **username** (*str*) – The user name to use when connecting to the active directory services
- **password** (*str*) – The password to use when connecting to the active directory services
- **ou** (*str, optional*) – The OU path to use when connecting to the active directory services, defaults to None
- **check\_connection** (*bool, optional*) – Check connectivity before attempting to connect to directory services, defaults to *False*

```
filer.directoryservice.connect('ctera.local', 'administrator', 'B4tMob!13')

"""Connect to the EMEA Organizational Unit"""

filer.directoryservice.connect('ctera.local', 'administrator', 'B4tMob!13', 'ou=EMEA,
↪dc=ctera, dc=local')
```

---

**Note:** the *ou* parameter must specify the distinguished name of the organizational unit

---

`DirectoryService.advanced_mapping(domain, start, end)`

Configure advanced mapping

#### Parameters

- **domain** (*str*) – The active directory domain
- **start** (*int*) – The minimum id to use for mapping
- **end** (*int*) – The maximum id to use for mapping

```
filer.directoryservice.advanced_mapping('CTERA', 200001, 5000001)
```

---

**Note:** to retrieve a list of domain flat names, use `cterasdk.edge.directoryservice.domains()`

---

`DirectoryService.disconnect()`

Disconnect from Active Directory Service

```
filer.directoryservice.disconnect()
```

`DirectoryService.domains()`

Get all domains

**Return list(str)** List of names of all discovered domains

```
domains = filer.directoryservice.domains()

print(domains)
```

DirectoryService.**set\_static\_domain\_controller**(*dc*)

Configure the Gateway to use a static domain controller

**Parameters** *dc* (*str*) – The FQDN, hostname or ip address of the domain controller

**Returns** The FQDN, hostname or ip address of the domain controller

**Return type** *str*

```
filer.directoryservice.set_static_domain_controller('192.168.90.1')
```

DirectoryService.**get\_static\_domain\_controller**()

Retrieve the static domain controller configuration

**Returns** A FQDN, hostname or ip address of the domain controller

**Return type** *str*

```
domain_controller = filer.directoryservice.get_static_domain_controller()
print(domain_controller)
```

DirectoryService.**remove\_static\_domain\_controller**()

Delete the static domain controller configuration

```
filer.directoryservice.remove_static_domain_controller()
```

## Cloud Services

Services.**connect**(*server*, *user*, *password*, *ctera\_license='EV16'*)

Connect to a Portal.

**The connect method will first validate the *license* argument**, ensure the Gateway can establish a TCP connection over port 995 to *server* using Gateway.tcp\_connect() and verify the Portal does not require device activation via code

### Parameters

- **server** (*str*) – Address of the Portal
- **user** (*str*) – User for the Portal connection
- **password** (*str*) – Password for the Portal connection
- **ctera\_license** (*cterasdk.edge.enum.License, optional*) – CTERA License, defaults to cterasdk.edge.enum.License.EV16

**Warning:** for any certificate related error, this library will prompt for your consent in order to proceed. to avoid the prompt, you may configure *chopin-core* to automatically trust the server's certificate, using: `config.connect['ssl'] = 'Trust'`

```
filer.services.connect('chopin.ctera.com', 'svc_account', 'Th3AmazingR4ce!', 'EV32') #_
↳ activate as an EV32
```

```
filer.services.connect('52.204.15.122', 'svc_account', 'Th3AmazingR4ce!', 'EV64') #_
↳ activate as an EV64
```

`Services.activate(server, user, code, ctera_license='EV16')`

Activate the gateway using an activation code

### Parameters

- **server** (*str*) – Address of the Portal
- **user** (*str*) – User for the Portal connection
- **code** (*str*) – Activation code for the Portal connection
- **ctera\_license** (`cterasdk.edge.enum.License`, *optional*) – CTERA License, defaults to `cterasdk.edge.enum.License.EV16`

This method's behavior is identical to `cterasdk.edge.services.Services.connect()`

```
filer.services.activate('chopin.ctera.com', 'svc_account', 'fd3a-301b-88d5-e1a9-cbdb') #  
↪ activate as an EV16
```

`Services.reconnect()`

Reconnect to the Portal

```
filer.services.reconnect()
```

`Services.disconnect()`

Disconnect from the Portal

```
filer.services.disconnect()
```

`Services.enable_sso()`

Enable SSO connection

## Applying a License

`Licenses.apply(ctera_license)`

Apply a license

**Parameters** `ctera_license` (`cterasdk.edge.enum.License`) – License type

```
filer.license.apply('EV32')
```

---

**Note:** you can specify a license upon connecting the Gateway to CTERA Portal. See `cterasdk.edge.services.Services.connect()`

---

## Caching

`Cache.enable()`

Enable caching

```
filer.cache.enable()
```

`Cache.disable()`

Disable caching



```
filer.cache.disable()
```

**Warning:** all data synchronized from the cloud will be deleted and all unsynchronized changes will be lost.

Cache.**force\_eviction()**

Force eviction

```
filer.cache.force_eviction()
```

Cache.**pin**(*path*)

Pin a folder

**Parameters** *path* (*str*) – Directory path

```
""" Pin a cloud folder named 'data' owned by 'Service Account' """
filer.cache.pin('users/Service Account/data')
```

Cache.**pin\_exclude**(*path*)

Exclude a sub-folder from a pinned folder

**Parameters** *path* (*str*) – Directory path

```
""" Exclude a subfolder from a pinned cloud folder """
filer.cache.pin_exclude('users/Service Account/data/accounting')
```

Cache.**remove\_pin**(*path*)

Remove a pin from a previously pinned folder

**Parameters** *path* (*str*) – Directory path

```
""" Remove a pin from a previously pinned folder """
filer.cache.remove_pin('users/Service Account/data')
```

Cache.**pin\_all**()

Pin all folders

```
""" Pin all folders """
filer.cache.pin_all()
```

Cache.**unpin\_all**()

Remove all folder pins

```
""" Remove all folder pins """
filer.cache.unpin_all()
```

### Cloud Backup

`Backup.configure(passphrase=None)`  
Gateway backup configuration

**Parameters** `passphrase` (*str, optional*) – Passphrase for the backup, defaults to None

```
"""Configure backup without a passphrase"""
```

```
filer.backup.configure()
```

`Backup.start()`  
Start backup

```
filer.backup.start()
```

`Backup.suspend()`  
Suspend backup

```
filer.backup.suspend()
```

`Backup.unsuspend()`  
Unsuspend backup

```
filer.backup.unsuspend()
```

### Cloud Backup Files

`BackupFiles.unselect_all()`  
Unselect all files from backup

```
filer.backup.files.unselect_all()
```

### Cloud Sync

`Sync.suspend(wait=True)`  
Suspend Cloud Sync

**Parameters** `wait` (*bool*) – Wait for synchronization to stop

```
filer.sync.suspend()
```

`Sync.unsuspend()`  
Unsuspend Cloud Sync

```
filer.sync.unsuspend()
```

`Sync.exclude_files(extensions=None, filenames=None, paths=None, custom_exclusion_rules=None)`  
Exclude files from Cloud Sync. This method will override any existing file exclusion rules Use `cterasdk.common.types.FileFilterBuilder()` to build custom file exclusion rules`

**Parameters**

- **extensions** (*list[str]*) – List of file extensions

- **filenames** (*list[str]*) – List of file names
- **paths** (*list[str]*) – List of file paths
- **rules** (*list[cterasdk.common.types.FilterBackupSet]*) – Set of custom exclusion rules

```

filer.sync.exclude_files(['exe', 'cmd', 'bat']) # exclude file extensions

filer.sync.exclude_files(filenames=['Cloud Sync.lnk', 'The quick brown fox.docx']) #_
↳exclude file names

"""Exclude file extensions and file names"""
filer.sync.exclude_files(['exe', 'cmd'], ['Cloud Sync.lnk'])

"""
Create a custom exclusion rule

Exclude files that their name starts with 'tmp' and smaller than 1 MB (1,048,576 bytes)
"""
name_filter_rule = common_types.FileFilterBuilder.name().startswith('tmp')
size_filter_rule = common_types.FileFilterBuilder.size().less_than(1048576)
exclusion_rule = common_types.FilterBackupSet('Custom exclusion rule', filter_
↳rules=[name_filter_rule, size_filter_rule])

filer.sync.exclude_files(custom_exclusion_rules=[exclusion_rule])

```

**Sync.remove\_file\_exclusion\_rules()**  
Remove previously configured sync exclusion rules

```
filer.sync.remove_file_exclusion_rules()
```

## Cloud Sync Bandwidth Throttling

**CloudSyncBandwidthThrottling.get\_policy()**  
Get the bandwidth throttling policy

**Returns** a list of bandwidth throttling rules

**Return type** *list[cterasdk.common.types.ThrottlingRule]*

**CloudSyncBandwidthThrottling.set\_policy(rules)**  
Set the bandwidth throttling policy

**Parameters rules** (*list[cterasdk.common.types.ThrottlingRule]*) – List of bandwidth throttling rules

```

"""Throttle bandwidth during business hours on week days: Monday - Friday"""
schedule1 = common_types.TimeRange().start('07:00:00').end('19:00:00').days(common_enum.
↳DayOfWeek.Weekdays).build()
rule1 = common_types.ThrottlingRuleBuilder().upload(50).download(50).schedule(schedule1).
↳build()

"""Throttle bandwidth off business hours on week days: Monday - Friday"""
schedule2 = common_types.TimeRange().start('19:00:00').end('07:00:00').days(common_enum.
↳DayOfWeek.Weekdays).build()

```

(continues on next page)

(continued from previous page)

```
rule2 = common_types.ThrottlingRuleBuilder().upload(100).download(100).
↳schedule(schedule2).build()

"""Throttle bandwidth during weekends: Saturday, Sunday"""
schedule3 = common_types.TimeRange().start('00:00:00').end('23:59:00').days(common_enum.
↳DayOfWeek.Weekend).build()
rule3 = common_types.ThrottlingRuleBuilder().upload(500).download(500).
↳schedule(schedule3).build()

filer.sync.throttling.set_policy([rule1, rule2, rule3])
```

## File Access Protocols

**FTP.disable()**  
Disable FTP

```
filer.ftp.disable()
```

**AFP.disable()**  
Disable AFP

```
filer.afp.disable()
```

**NFS.disable()**  
Disable NFS

```
filer.nfs.disable()
```

**RSync.disable()**  
Disable FTP

```
filer.rsync.disable()
```

## Windows File Sharing (CIFS/SMB)

**SMB.enable()**  
Enable SMB

```
filer.smb.enable()
```

**SMB.disable()**  
Disable SMB

```
filer.smb.disable()
```

**SMB.set\_packet\_signing(packet\_signing)**  
Set Packet signing

**Parameters packet\_signing** (`cterasdk.edge.enum.CIFSPacketSigning`) – Packet signing type

```
filer.smb.set_packet_signing('If client agrees')
```

SMB.**enable\_abe**()  
Enable ABE

```
filer.smb.enable_abe()
```

SMB.**disable\_abe**()  
Disable ABE

```
filer.smb.disable_abe()
```

AIO.**enable**()  
Enable AIO

```
filer.aio.enable()
```

AIO.**disable**()  
Disable AIO

```
filer.aio.disable()
```

## Network

Network.**set\_static\_ipaddr**(*address, subnet, gateway, primary\_dns\_server, secondary\_dns\_server=None*)  
Set a Static IP Address

### Parameters

- **address** (*str*) – The static address
- **subnet** (*str*) – The subnet for the static address
- **gateway** (*str*) – The default gateway
- **primary\_dns\_server** (*str*) – The primary DNS server
- **secondary\_dns\_server** (*str, optional*) – The secondary DNS server, defaults to None

```
filer.network.set_static_ipaddr('10.100.102.4', '255.255.255.0', '10.100.102.1', '10.100.102.1')
```

```
filer.show('/status/network/ports/0/ip') # will print the IP configuration
```

Network.**set\_static\_nameserver**(*primary\_dns\_server, secondary\_dns\_server=None*)  
Set the DNS Server addresses statically

### Parameters

- **primary\_dns\_server** (*str*) – The primary DNS server
- **secondary\_dns\_server** (*str, optional*) – The secondary DNS server, defaults to None

```
filer.network.set_static_nameserver('10.100.102.1') # to set the primary name server
```

```
filer.network.set_static_nameserver('10.100.102.1', '10.100.102.254') # to set both_
↳ primary and secondary
```

`Network.enable_dhcp()`  
Enable DHCP

```
filer.network.enable_dhcp()
```

`Network.set_mtu(mtu)`  
Set a custom network maximum transmission unit (MTU)

**Parameters** `mtu (int)` – Maximum transmission unit

```
filer.network.set_mtu(1320) # set the maximum transmission unit (MTU) to 1320
filer.network.set_mtu(9000) # configure 'jumbo' frames (MTU: 9000)
```

`Network.reset_mtu()`  
Set the default maximum transmission unit (MTU) settings

```
filer.network.reset_mtu() # disable custom mtu configuration and restore default_
↪setting (1500)
```

## Network Diagnostics

`Network.tcp_connect(service)`  
Test a TCP connection between the Gateway and the provided host address

**Parameters** `service (cterasdk.edge.types.TCPService)` – A service, identified by a host and a port

**Returns** A named-tuple including the host, port and a boolean value indicating whether TCP connection can be established

**Return type** `cterasdk.edge.types.TCPConnectResult`

```
cttp_service = gateway_types.TCPService('chopin.ctera.com', 995)
result = filer.network.tcp_connect(cttp_service)
if result.is_open:
    print('Success')
    # do something...
else:
    print('Failure')

ldap_service = gateway_types.TCPService('dc.ctera.com', 389)
filer.network.tcp_connect(ldap_service)
```

`Network.diagnose(services)`  
Test a TCP connection to a host over a designated port

**Parameters** `services (list[cterasdk.edge.types.TCPService])` – List of services, identified by a host and a port

**Returns** A list of named-tuples including the host, port and a boolean value indicating whether TCP connection can be established

**Return type** `list[cterasdk.edge.types.TCPConnectResult]`

```

services = []
services.append(gateway_types.TCPService('192.168.90.1', 389)) # LDAP
services.append(gateway_types.TCPService('ctera.portal.com', 995)) # CTTTP
services.append(gateway_types.TCPService('ctera.portal.com', 443)) # HTTPS
result = filer.network.diagnose(services)
for result in results:
    print(result.host, result.port, result.is_open)

```

`Network.iperf(address, port=5201, threads=1, protocol='TCP', direction='Upload', retries=120, seconds=1)`  
 Invoke a network throughput test

#### Parameters

- **address** (*str*) – The host running the iperf server
- **port** (*int, optional*) – The iperf server port, defaults to 5201
- **threads** (*int, optional*) – The number of threads, defaults to 1
- **protocol** (`cterasdk.edge.enum.IPProtocol`, *optional*) – IP protocol, defaults to 'TCP'
- **direction** (`cterasdk.edge.enum.Traffic`, *optional*) – Traffic direction, defaults to 'Upload'
- **retries** (*int, optional*) – Number of retries when sampling the iperf task status, defaults to 120
- **seconds** (*int, optional*) – Number of seconds to wait between retries, defaults to 1

**Returns** A string containing the iperf output

**Return type** `str`

```

filer.network.iperf('192.168.1.145') # iperf server: 192.168.1.145, threads: 1, measure_
↪upload over TCP port 5201

filer.network.iperf('192.168.1.145', port=85201, threads=5) # Customized port and_
↪number of threads

filer.network.iperf('192.168.1.145', direction=gateway_enum.Traffic.Download) # Measure_
↪download speed

filer.network.iperf('192.168.1.145', protocol=gateway_enum.IPProtocol.UDP) # Use UDP

```

## Mail Server

`Mail.enable(smtp_server, port=25, username=None, password=None, use_tls=True)`  
 Enable e-mail delivery using a custom SMTP server

#### Parameters

- **smtp\_server** (*str*) – Address of the SMTP Server
- **port** (*int, optional*) – The listening port of the SMTP Server, defaults to 25
- **username** (*str, optional*) – The user name of the SMTP Server, defaults to None
- **password** (*str, optional*) – The password of the SMTP Server, defaults to None

- **use\_tls** (*bool, optional*) – Use TLS when connecting to the SMTP Server, defaults to True

```
filer.mail.enable('smtp.ctera.com') # default settings

filer.mail.enable('smtp.ctera.com', 465) # custom port number

"""Use default port number, use authentication and require TLS"""

filer.mail.enable('smtp.ctera.com', username = 'user', password = 'secret', useTLS = True)
```

Mail.**disable**()

Disable e-mail delivery using a custom SMTP server

```
filer.mail.disable()
```

## Logging

Logs.**settings**(*retention, min\_severity=None*)

Configure log settings

### Parameters

- **retention** (*int*) – Log retention period in days
- **min\_severity** (*cterasdk.edge.enum.Severity, optional*) – Minimal log severity

Logs.**logs**(*topic, include=None, minSeverity='info'*)

Fetch Gateway logs

### Parameters

- **topic** (*str*) – Log Topic to fetch
- **include** (*list[str], optional*) – List of fields to include in the response, defaults to Logs.default\_include
- **minSeverity** (*cterasdk.edge.enum.Severity, optional*) – Minimal log severity to fetch, defaults to cterask.edge.enum.Severity.INFO

**Returns** Log lines

**Return type** *cterasdk.lib.iterator.Iterator*

Syslog.**enable**(*server, port=514, proto='UDP', min\_severity='info'*)

Enable Syslog

### Parameters

- **server** (*str*) – Server address to send syslog logs
- **port** (*int, optional*) – Syslog server communication port, defaults to 514
- **proto** (*cterasdk.edge.enum.IPProtocol, optional*) – Syslog server communication protocol, defaults to cterask.edge.enum.IPProtocol.UDP
- **min\_severity** (*cterasdk.edge.enum.Severity, optional*) – Minimal log severity to fetch, defaults to cterask.edge.enum.Severity.INFO



```
filer.syslog.enable('syslog.ctera.com') # default settings

filer.syslog.enable('syslog.ctera.com', proto = 'TCP') # use TCP

filer.syslog.enable('syslog.ctera.com', 614, minSeverity = 'error') # use 614 UDP, ↵
↳severity >= error
```

**Syslog.disable()**  
Disable Syslog

```
filer.syslog.disable()
```

## SMB Audit Logs

**Audit.enable**(*path*, *auditEvents=None*, *logKeepPeriod=30*, *maxLogKBSize=102400*, *maxRotateTime=1440*, *includeAuditLogTag=True*, *humanReadableAuditLog=False*)

Enable Gateway Audit log

### Parameters

- **path** (*str*) – Path to save the audit log
- **auditEvents** (*list[cterasdk.edge.enum.AuditEvents]*, *optional*) – List of audit event types to save, defaults to `Audit.defaultAuditEvents`
- **logKeepPeriod** (*int*, *optional*) – Period to key the logs in days, defaults to 30
- **maxLogKBSize** (*int*, *optional*) – The maximum size of the log file in KB, defaults to 102400 (100 MB)
- **maxRotateTime** (*int*, *optional*) – The maximal time before rotating the log file in Minutes, defaults to 1440 (24 hours)
- **includeAuditLogTag** (*bool*, *optional*) – Include audit log tag, defaults to True
- **humanReadableAuditLog** (*bool*, *optional*) – Human readable audit log, defaults to False

```
filer.audit.enable('/logs')
```

**Audit.disable()**  
Disable Gateway Audit log

```
filer.audit.disable()
```

## Reset

**Power.reset**(*wait=False*)  
Reset the Gateway setting

**Parameters** **wait** (*bool*, *optional*) – Wait got the reset to complete, defaults to False

```
filer.power.reset() # will reset and immediately return

filer.power.reset(True) # will reset and wait for the Gateway to boot
```

### See also:

create the first admin account after resetting the Gateway to its default settings: `cterasdk.edge.users.Users.add_first_user()`

## SSL

### SSL.`disable_http()`

Disable HTTP access

```
filer.ssl.disable_http()
```

### SSL.`enable_http()`

Enable HTTP access

```
filer.ssl.enable_http()
```

### SSL.`is_http_disabled()`

Check if HTTP access is disabled

```
filer.ssl.is_http_disabled()
```

### SSL.`is_http_enabled()`

Check if HTTP access is enabled

```
filer.ssl.is_http_enabled()
```

```
"""
ca_certificate = './certs/certificate.crt'
"""

filer.ssl.set_trusted_ca(ca_certificate)
```

### SSL.`get_storage_ca()`

Get object storage trusted CA certificate

### SSL.`remove_storage_ca()`

Remove object storage trusted CA certificate

### SSL.`import_certificate(private_key, *certificates)`

Import the Edge Filer's web server's SSL certificate

#### Parameters

- **private\_key** (*str*) – The PEM-encoded private key, or a path to the PEM-encoded private key file
- **certificates** (*list[str]*) – The PEM-encoded certificates, or a list of paths to the PEM-encoded certificates

```
"""
certificate = './certs/certificate.crt'
intermediate_cert = './certs/certificate1.crt'
ca_certificate = './certs/certificate2.crt'
private_key = './certs/private.key'
"""
```

(continues on next page)

(continued from previous page)

```

"""
Specify certificates in the following order: domain cert, intermediary certs, CA cert
You may include as many intermediate certificates as needed
"""
filer.ssl.import_certificate(private_key, certificate, intermediate_cert, ca_certificate)

```

## Power Management

`Power.reboot(wait=False)`

Reboot the Gateway

**Parameters** `wait (bool, optional)` – Wait got the reboot to complete, defaults to False

```
filer.power.reboot() # will reboot and immediately return
```

```
filer.power.reboot(True) # will reboot and wait
```

`Power.shutdown()`

Shutdown the Gateway

```
filer.power.shutdown()
```

## SNMP

`SNMP.is_enabled()`

Check if SNMP is enabled

**Returns** True is SNMP is enabled, else False

**Return type** bool

```
filer.snmp.is_enabled()
```

`SNMP.enable(port=161, community_str=None, username=None, password=None)`

Enable SNMP

**Parameters**

- **port** (*int, optional*) – SNMP server port, defaults to 161
- **community\_str** (*str, optional*) – SNMPv2c community string
- **username** (*str, optional*) – SNMPv3 username
- **password** (*str, optional*) – SNMPv3 password

```
filer.snmp.enable(community_str='MpPcK12sArSdTLZ4URj4')
```

`SNMP.disable()`

Disable SNMP

```
filer.snmp.disable()
```

`SNMP.modify(port=None, community_str=None, username=None, password=None)`

Modify current SNMP configuration. Only configurations that are not *None* will be changed. SNMP must be enabled

### Parameters

- **port** (*int, optional*) – SNMP server port, defaults to 161
- **community\_str** (*str, optional*) – SNMPv2c community string
- **username** (*str, optional*) – SNMPv3 username
- **password** (*str, optional*) – SNMPv3 password

```
filer.snmp.modify(community_str='L0K2zGpgmOQH2CXaUSuB', username='snmp_user', password='gVQBaHSOGV')
```

`SNMP.get_configuration()`

`filer.snmp.get_configuration()`

## Support

### Support Report

`Support.get_support_report()`

Download support report

## Debug

`Support.set_debug_level(*levels)`

Set the debug level

```
filer.support.set_debug_level('backup', 'process', 'cttp', 'samba')
```

```
filer.support.set_debug_level('info')
```

```
filer.support.set_debug_level('caching', 'evictor')
```

## Telnet Access

`Telnet.enable(code)`

Enable Telnet

```
filer.telnet.enable('a7df639a')
```

`Telnet.disable()`

Disable Telnet

```
filer.telnet.disable()
```

## SSH Access

SSH.**enable**(*public\_key=None, public\_key\_file=None, exponent=65537, key\_size=2048*)

Enable the Edge Filer's SSH daemon

### Parameters

- **public\_key** (*str, optional*) – A PEM-encoded public key in OpenSSH format. If neither a public key nor public key file were specified, an RSA key pair will be generated automatically. The PEM-encoded private key will be saved to the default Downloads directory
- **public\_key\_file** (*str, optional*) – A path to the public key file
- **exponent** (*int, optional*) – The public exponent of the new key, defaults to *65537*
- **key\_size** (*int, optional*) – The length of the modulus in bits, defaults to *2048*

```

"""Enable SSH access"""
filer.ssh.enable()

"""Enable SSH access using a public key file"""
filer.ssh.enable(public_key_file='./public_key.pub') # relative to the current directory
filer.ssh.enable(public_key_file='C:\\Users\\jsmith\\Desktop\\public_key.pub') # full
↳path

"""Generate an RSA key pair and enable SSH access"""

from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.serialization import Encoding, PrivateFormat,
↳PublicFormat, NoEncryption

private_key = rsa.generate_private_key(public_exponent=exponent, key_size=key_size)
public_key = private_key.public_key().public_bytes(Encoding.OpenSSH, PublicFormat.
↳OpenSSH).decode('utf-8')

filer.ssh.enable(public_key)

"""Print PEM-encoded RSA private key"""
print(private_key.private_bytes(Encoding.PEM, PrivateFormat.OpenSSH, NoEncryption()).
↳decode('utf-8'))

"""Print OpenSSH formatted RSA public key"""
print(public_key)

```

SSH.**disable**()

```
filer.ssh.disable()
```

## 2.2.2 File Browser

### Table of Contents

- *File Browser*
  - *Obtaining Access to the Gateway's File System*
    - \* *List*
    - \* *Download*
    - \* *Create Directory*
    - \* *Copy*
    - \* *Move*
    - \* *Delete*

### 2.2.2.1 Obtaining Access to the Gateway's File System

```
filer = Gateway('vGateway-0dbc')
filer.login('USERNAME', 'PASSWORD')
file_browser = filer.files # the field is an instance of FileBrowser class object
```

#### List

```
static FileBrowser.ls(_path)
```

#### Download

```
FileBrowser.download(path, destination=None)
Download a file
```

##### Parameters

- **path** (*str*) – The file path on the Edge Filer
- **destination** (*str, optional*) – File destination, if it is a directory, the original filename will be kept, defaults to the default directory

```
file_browser.download('cloud/users/Service Account/My Files/Documents/Sample.docx')
```

## Create Directory

`FileBrowser.mkdir(path, recurse=False)`

Create a new directory

### Parameters

- **path** (*str*) – The path of the new directory
- **recurse** (*bool, optional*) – Create subdirectories if missing, defaults to False

```
file_browser.mkdir('cloud/users/Service Account/My Files/Documents')

file_browser.mkdir('cloud/users/Service Account/My Files/The/quick/brown/fox', recurse =
↳ True)
```

## Copy

`FileBrowser.copy(src, dest, overwrite=False)`

Copy a file or a folder

### Parameters

- **src** (*str*) – Source file or folder path
- **dest** (*str*) – Destination folder path
- **overwrite** (*bool, optional*) – Overwrite on conflict, defaults to False

```
"""
Copy the 'Documents' folder from Bruce Wayne to Alice Wonderland
The full path of the documents folder after the copy: 'cloud/users/Alice Wonderland/My_
↳ Files/Documents'
"""
file_browser.copy('cloud/users/Bruce Wayne/My Files/Documents', 'cloud/users/Alice_
↳ Wonderland/My Files')

"""Copy the file Summary.xlsx to another directory, and overwrite on conflict"""
file_browser.copy('cloud/users/Bruce Wayne/My Files/Summary.xlsx', 'cloud/users/Bruce_
↳ Wayne/Spreadsheets', True)
```

## Move

`FileBrowser.move(src, dest, overwrite=False)`

Move a file or a folder

### Parameters

- **src** (*str*) – Source file or folder path
- **dest** (*str*) – Destination folder path
- **overwrite** (*bool, optional*) – Overwrite on conflict, defaults to False

```

"""
Move the 'Documents' folder from Bruce Wayne to Alice Wonderland
The full path of the documents folder after the move: 'cloud/users/Alice_
↳Files/Documents'
"""
file_browser.move('cloud/users/Bruce Wayne/My Files/Documents', 'cloud/users/Alice_
↳Wonderland/My Files')

"""Move the file Summary.xlsx to another directory, and overwrite on conflict"""
file_browser.move('cloud/users/Bruce Wayne/My Files/Summary.xlsx', 'cloud/users/Bruce_
↳Wayne/Spreadsheets', True)

```

## Delete

`FileBrowser.delete(path)`

Delete a file

**Parameters** `path (str)` – The file's path on the gateway

```
file_browser.delete('cloud/users/Service Account/My Files/Documents')
```

## 2.3 Agent

**class** `cterasdk.object.Agent.Agent(host, port=80, https=False, Portal=None)`

Main class operating on a Agent

### Variables

- **backup** (`cterasdk.edge.backup.Backup`) – Object holding the Agent Backup APIs
- **cli** (`cterasdk.edge.cli.CLI`) – Object holding the Agent CLI APIs
- **logs** (`cterasdk.edge.logs.Logs`) – Object holding the Agent Logs APIs
- **services** (`cterasdk.edge.services.Services`) – Object holding the Agent Services APIs
- **support** (`cterasdk.edge.support.Support`) – Object holding the Agent Support APIs
- **sync** (`cterasdk.edge.sync.Sync`) – Object holding the Agent Sync APIs

`__init__(host, port=80, https=False, Portal=None)`

### Parameters

- **host** (`str`) – The fully qualified domain name, hostname or an IPv4 address of the Gateway
- **port** (`int, optional`) – Set a custom port number (0 - 65535), defaults to 80
- **https** (`bool, optional`) – Set to True to require HTTPS, defaults to False
- **Portal** (`cterasdk.object.Portal.Portal, optional`) – The portal through which the remote session was created, defaults to None



## 2.4 Miscellaneous

### 2.4.1 Logging

The library includes a built-in console logger. The logger's configuration is controlled by the `config.Logging.get()` class object.

#### 2.4.1.1 Redirecting the log to a file

You can redirect the cterasdk log to a file by setting the environment variable `CTERASDK_LOG_FILE`

#### 2.4.1.2 Disabling the Logger

The logger is enabled by default. To disable the logger, run:

```
config.Logging.get().disable()
```

#### 2.4.1.3 Changing the Log Level

The default logging level is set to `logging.INFO`. To change the log level, run:

```
config.Logging.get().setLevel(logging.ERROR) # will log severity >= error
config.Logging.get().setLevel(logging.WARNING) # will log severity >= warning
```

### Log Levels

The available log levels are:

Level	Numeric Value
CRITICAL	50
ERROR	40
WARNING	30
INFO	20
DEBUG	10

### 2.4.2 Formatting

The following formatting functions are included in this library:

`cterasdk.convert.format.tojsonstr(obj, pretty_print=True, no_log=True)`

Convert a Python object to a JSON string.

#### Parameters

- **obj** (*object*) – the Python object
- **pretty\_print** (*bool*) – Whether to format the JSON string, defaults to True
- **no\_log** (*bool*) – Hide sensitive values in the log messages

**Returns** JSON string of the object

**Return type** str

```
user = Object()
user.name = 'alice'
user.firstName = 'Alice'
user.lastName = 'Wonderland'
user.email = 'alice@adventures.com'
user.password = 'Passw0rd1!'
print(tojsonstr(user))
{
  "lastName": "Wonderland",
  "password": "Passw0rd1!",
  "name": "alice",
  "firstName": "Alice",
  "email": "alice@adventures.com"
}
print(tojsonstr(user, False))
{"lastName": "Wonderland", "password": "Passw0rd1!", "name": "alice", "firstName": "Alice
↵", "email": "alice@adventures.com"}
```

`cterasdk.convert.format.toxmlstr(obj, pretty_print=False)`  
Convert a Python object to an XML string

**Parameters**

- **obj** (*object*) – the Python object
- **pretty\_print** (*bool*) – whether to format the XML string, defaults to False

**Returns** XML string of the object

**Return type** str

```
user = Object()
user.name = 'alice'
user.firstName = 'Alice'
user.lastName = 'Wonderland'
user.email = 'alice@adventures.com'
user.password = 'Passw0rd1!'
print(toxmlstr(user))
print(toxmlstr(user, True))
```

## CTERASDK PACKAGE

### 3.1 Subpackages

#### 3.1.1 cterasdk.client package

##### 3.1.1.1 Submodules

##### cterasdk.client.cteraclient module

```
class cterasdk.client.cteraclient.CTERAClient(session_id_key)
    Bases: object
    copy(baseurl, src, dest, overwrite)
    db(baseurl, path, name, param)
    delete(baseurl, path)
    download(baseurl, path, params)
    download_zip(baseurl, path, form_data)
    execute(baseurl, path, name, param=None)
    static file_descriptor(request, response)
    form_data(baseurl, path, form_data)
    static fromxmlstr(request, response)
    get(baseurl, path, params=None)
    get_multi(baseurl, path, paths)
    get_session_id()
    mkcol(baseurl, path)
    move(baseurl, src, dest, overwrite)
    multipart(baseurl, path, form_data)
    post(baseurl, path, data)
    put(baseurl, path, data)
    set_authorization_headers(headers)
    set_session_id(session_id)
```

`upload(baseurl, path, form_data)`

### `cterasdk.client.host` module

**class** `cterasdk.client.host.CTERAHost`(*host, port, https*)

Bases: `cterasdk.client.host.NetworkHost`

**add**(*path, param, use\_file\_url=False*)

Add a schema object.

**property** `base_api_url`

**property** `base_file_url`

**copy**(*src, dest, overwrite, use\_file\_url=False*)

**db**(*path, name, param, use\_file\_url=False*)

**default\_class**(*name*)

**delete**(*path, use\_file\_url=False*)

Delete a schema object.

**download\_zip**(*path, form\_data, use\_file\_url=False*)

**execute**(*path, name, param=None, use\_file\_url=False*)

Execute a schema object method.

**form\_data**(*path, form\_data, use\_file\_url=False*)

**get**(*path, params=None, use\_file\_url=False*)

Retrieve a schema object as a Python object.

**get\_multi**(*path, paths, use\_file\_url=False*)

Retrieve one or more schema objects as a Python object.

**get\_session\_id**()

Get the id of the current session

**Return str** Current session id

**login**(*username, password*)

Log in

#### Parameters

- **username** (*str*) – User name to log in
- **password** (*str*) – User password

**logout**()

Log out

**mkcol**(*path, use\_file\_url=False*)

**move**(*src, dest, overwrite, use\_file\_url=False*)

**multipart**(*path, form\_data, use\_file\_url=False*)

**openfile**(*path, params=None, use\_file\_url=False*)

**post**(*path, value, use\_file\_url=False*)

**put**(*path, value, use\_file\_url=False*)

Update a schema object or attribute.

**register\_session**(*session*)

**session**()

**set\_authorization\_headers**(*headers*)

Start a session using authorization headers id instead of logging in

**Parameters headers** (*dict*) – the authorization headers, represented as a key-value str dict

**set\_session\_id**(*session\_id*)

Start a session with the session id instead of logging in

**Parameters session\_id** (*str*) – Session id for the new session

**show**(*path, use\_file\_url=False*)

Print a schema object as a JSON string.

**show\_multi**(*path, paths, use\_file\_url=False*)

Print one or more schema objects as a JSON string.

**upload**(*path, form\_data, use\_file\_url=False*)

**whoami**()

Return the name of the logged in user.

**Return cterasdk.common.object.Object** The session object of the current user

**class** cterasdk.client.host.**NetworkHost**(*host, port, https*)

Bases: object

**baseurl**()

**host**()

**https**()

**port**()

**scheme**()

**test\_conn**()

cterasdk.client.host.**authenticated**(*function*)

### cterasdk.client.http module

**class** cterasdk.client.http.**ContentType**

Bases: object

**textplain** = {'Content-Type': 'text/plain'}

**urlencoded** = {'Content-Type': 'application/x-www-form-urlencoded'}

**class** cterasdk.client.http.**HTTPClient**(*session\_id\_key*)

Bases: [cterasdk.client.http.HttpClientBase](#)

**copy**(*src, dest, overwrite, headers=None*)

**delete**(*url, headers=None*)

**get**(*url, params=None, headers=None, stream=None*)

**mkcol**(*url, headers=None*)

**move**(*src, dest, overwrite, headers=None*)

```
multipart(url, form_data, monitor_function_generator=None)
post(url, headers=None, data="", urlencode=False)
put(url, headers=None, data="")
upload(url, form_data)

exception cterask.client.http.HTTPException(http_error)
    Bases: Exception

class cterask.client.http.HTTPResponse(response)
    Bases: object

    getcode()
    geturl()
    read()

class cterask.client.http.HttpClientBase(session_id_key)
    Bases: object

    dispatch(ctera_request)
    get_session_id()
    on_ssl_error(request)
    static on_timeout(attempt)
    set_custom_headers(headers)
        Add custom headers that will be included in every http request.

        Parameters headers (dict) – the headers, represented as a key-value str dict
    set_session_id(session_id)
    should_trust(host, port)
    trust(_host, _port)

class cterask.client.http.HttpClientRequest(method, url, **kwargs)
    Bases: object

class cterask.client.http.HttpClientRequestCopy(src, dest, overwrite, headers=None)
    Bases: cterasdk.client.http.HttpClientRequestCopyMove

class cterask.client.http.HttpClientRequestCopyMove(method, src, dest, overwrite, headers=None)
    Bases: cterasdk.client.http.HttpClientRequest

class cterask.client.http.HttpClientRequestDelete(url, headers=None)
    Bases: cterasdk.client.http.HttpClientRequest

class cterask.client.http.HttpClientRequestGet(url, params=None, headers=None, stream=None)
    Bases: cterasdk.client.http.HttpClientRequest

class cterask.client.http.HttpClientRequestMkcol(url, headers=None)
    Bases: cterasdk.client.http.HttpClientRequest

class cterask.client.http.HttpClientRequestMove(src, dest, overwrite, headers=None)
    Bases: cterasdk.client.http.HttpClientRequestCopyMove

class cterask.client.http.HttpClientRequestPost(url, headers=None, data=None)
    Bases: cterasdk.client.http.HttpClientRequest
```

**class** `cterasdk.client.http.HttpClientRequestPut`(*url, headers=None, data=None*)  
 Bases: `cterasdk.client.http.HttpClientRequest`  
`cterasdk.client.http.geturi`(*baseurl, path*)

### cterasdk.client.ssl module

**class** `cterasdk.client.ssl.CertificateServices`  
 Bases: `object`  
**static** `add_trusted_cert`(*host, port*)  
**static** `save_cert_from_server`(*host, port*)

## 3.1.2 cterasdk.common package

### 3.1.2.1 Submodules

#### cterasdk.common.datetime\_utils module

**class** `cterasdk.common.datetime_utils.DateTimeUtils`  
 Bases: `object`  
**static** `get_expiration_date`(*expiration*)  
 Get a `datetime.date` representation of the expiration date  
**Parameters** `expiration` (*variable, optional*) – The expiration value. Pass `datetime.date` for a specific date, integer for days from now, or `True` for immediate (yesterday)  
**Return** `datetime.date` `datetime.date` representation of the expiration date

#### cterasdk.common.item module

**class** `cterasdk.common.item.Item`  
 Bases: `object`

#### cterasdk.common.object module

**class** `cterasdk.common.object.Device`(*uid, version, firmware*)  
 Bases: `cterasdk.common.object.Object`

**class** `cterasdk.common.object.Object`  
 Bases: `object`

`cterasdk.common.object.delete_attr`(*obj, path*)  
 Delete attribute

#### Parameters

- **object** (`cterasdk.common.object.Object`) – The object
- **path** (*str*) – Attribute path

**Returns** The modified object

**Return type** `cterasdk.common.object.Object`

`cterasdk.common.object.delete_attrs(obj, paths)`

Delete attributes

**Parameters**

- **object** (`cterasdk.common.object.Object`) – The object
- **paths** (`list[str]`) – List of attributes to remove

**Returns** The modified object

**Return type** `cterasdk.common.object.Object`

`cterasdk.common.object.find_attr(obj, path)`

Find attribute

**Parameters**

- **object** (`cterasdk.common.object.Object`) – The object
- **path** (`str`) – A string or an array of the attribute path

**Returns** The attribute, or None if not found

`cterasdk.common.object.get_attr(obj, attr)`

Get attribute

**Parameters**

- **object** (`cterasdk.common.object.Object`) – The object
- **attr** (`str`) – The name of the attribute to retrieve

**Returns** The attribute, or None if not found

`cterasdk.common.object.remove_array_element(array, attr)`

`cterasdk.common.object.remove_array_element_by_key(array, key, value)`

`cterasdk.common.object.remove_attr(obj, attr)`

Remove attribute

**Parameters**

- **object** (`cterasdk.common.object.Object`) – The object
- **attr** (`str`) – The name of the attribute to remove

### `cterasdk.common.types` module

**class** `cterasdk.common.types.AdvancedFilterRule`(*classname, field, operator*)

Bases: `cterasdk.common.object.Object`

**class** `cterasdk.common.types.AfterOperator`(*right*)

Bases: `cterasdk.common.types.Operator`

**class** `cterasdk.common.types.ApplicationBackupSet`(*apps*)

Bases: `cterasdk.common.types.BackupSet`

**class** `cterasdk.common.types.BackupScheduleBuilder`

Bases: `object`

**static interval**(*hours=None, minutes=None*)

Schedule backup to periodically, defaults to 24 hours

**Parameters**



- **hours** (*int*) – Hours
- **minutes** (*int*) – Minutes

**static window**(*time\_range*)  
Schedule backup to run at a specific time

Parameters **time\_range** (`cterasdk.common.types.TimeRange`) – Time range

**class** `cterasdk.common.types.BackupSet`(*name, directory\_tree=None, filter\_rules=None, defaults\_dirs=None, template\_dirs=None, enabled=True, boolean\_function=None, comment=None*)

Bases: `cterasdk.common.object.Object`

**class** `cterasdk.common.types.BeforeOperator`(*right*)

Bases: `cterasdk.common.types.Operator`

**class** `cterasdk.common.types.BeginsWithOperator`(*right*)

Bases: `cterasdk.common.types.Operator`

**class** `cterasdk.common.types.ContainsOperator`(*right*)

Bases: `cterasdk.common.types.Operator`

**class** `cterasdk.common.types.CriteriaBuilder`(*criteria\_type, criteria\_field*)

Bases: `object`

**build**()

**class** `cterasdk.common.types.DateTimeCriteriaBuilder`(*criteria\_type, criteria\_field*)

Bases: `cterasdk.common.types.CriteriaBuilder`

**after**(*value*)

**before**(*value*)

**class** `cterasdk.common.types.DirEntry`(*name, display\_name=None, included=None, children=None*)

Bases: `cterasdk.common.types.FileEntry`

**class** `cterasdk.common.types.DirectoryEntryFactory`

Bases: `object`

**static root**(*included*)

**class** `cterasdk.common.types.EndsWithOperator`(*right*)

Bases: `cterasdk.common.types.Operator`

**class** `cterasdk.common.types.FileEntry`(*name, display\_name=None, included=None*)

Bases: `cterasdk.common.object.Object`

**class** `cterasdk.common.types.FileFilterBuilder`

Bases: `object`

**Type** = 'File'

**static extensions**()

Filter files by extension

**static last\_modified**()

Filter files by last modification date

**static name**()

Filter files by name pattern

**static names**()

Filter files by names

```
static path()  
    Filter files by path pattern  
static paths()  
    Filter files by path  
static size()  
    Filter files by size  
class cterasdk.common.types.FilterBackupSet(name, directory_tree=None, filter_rules=None,  
                                             defaults_dirs=None, template_dirs=None, enabled=True,  
                                             boolean_function=None, comment=None)  
  
    Bases: cterasdk.common.types.BackupSet  
class cterasdk.common.types.IntegerCriteriaBuilder(criteria_type, criteria_field)  
    Bases: cterasdk.common.types.CriteriaBuilder  
  
    less_than(value)  
    more_than(value)  
class cterasdk.common.types.IsOneOfOperator(right)  
    Bases: cterasdk.common.types.Operator  
class cterasdk.common.types.IsOperator(right)  
    Bases: cterasdk.common.types.Operator  
class cterasdk.common.types.LessThanOperator(right)  
    Bases: cterasdk.common.types.Operator  
class cterasdk.common.types.ListCriteriaBuilder(criteria_type, criteria_field)  
    Bases: cterasdk.common.types.CriteriaBuilder  
  
    include(values)  
class cterasdk.common.types.MoreThanOperator(right)  
    Bases: cterasdk.common.types.Operator  
class cterasdk.common.types.Operator(right)  
    Bases: cterasdk.common.object.Object  
class cterasdk.common.types.PolicyRule(assignment, criteria)  
    Bases: object  
class cterasdk.common.types.PolicyRuleConverter  
    Bases: object  
  
    static convert(rule, classname, property_name, assignment=None)  
class cterasdk.common.types.StringCriteriaBuilder(criteria_type, criteria_field)  
    Bases: cterasdk.common.types.CriteriaBuilder  
  
    contains(value)  
    endswith(value)  
    equals(value)  
    isoneof(values)  
    startswith(value)  
class cterasdk.common.types.TaskSchedule  
    Bases: cterasdk.common.object.Object
```

**class** `cterasdk.common.types.ThrottlingRule`

Bases: `object`

Throttling Rule

**Variables**

- **upload** (*int*) – Throttling rate upstream (Kilobits)
- **download** (*int*) – Throttling rate downstream (Kilobits)
- **start** (*str*) – Start time
- **end** (*str*) – End time
- **days** (*list[str]*) – Days

**static** `from_server_object(param)`

`to_server_object()`

**class** `cterasdk.common.types.ThrottlingRuleBuilder`

Bases: `object`

Bandwidth Throttling Rule Builder

**build**()

Build the throttling rule

**download**(*kbps*)

Throttle bandwidth downstream

**Parameters** **kbps** (*int*) – Kilobits per second

**schedule**(*schedule*)

Set the throttling rule schedule

**Parameters** **schedule** (`cterasdk.common.types.TimeRange`) – Schedule

**upload**(*kbps*)

Throttle bandwidth upstream

**Parameters** **kbps** (*int*) – Kilobits per second

**class** `cterasdk.common.types.TimeRange`

Bases: `object`

Class representing a task schedule

**build**()

Build the time range

**days**(*days*)

Set days

**Parameters** **days** (*list[cterasdk.common.enum.DayOfWeek]*) – A list of days

**end**(*end*)

End time

**Parameters** **end** (*str*) – A military time string ‘hh:mm:ss’ or a datetime object

**start**(*start*)

Start time

**Parameters** **start** (*str*) – A military time string ‘hh:mm:ss’ or a datetime object

`terminate_at_endtime()`

Terminate at end time, defaults to terminate on completion.

### 3.1.3 cterasdk.convert package

#### 3.1.3.1 Submodules

##### cterasdk.convert.exception module

**exception** `cterasdk.convert.exception.ParseException`

Bases: `Exception`

##### cterasdk.convert.format module

`cterasdk.convert.format.CreateElement`(*parent, tag*)

`cterasdk.convert.format.tojsonstr`(*obj, pretty\_print=True, no\_log=True*)

Convert a Python object to a JSON string.

###### Parameters

- **obj** (*object*) – the Python object
- **pretty\_print** (*bool*) – Whether to format the JSON string, defaults to `True`
- **no\_log** (*bool*) – Hide sensitive values in the log messages

**Returns** JSON string of the object

**Return type** `str`

`cterasdk.convert.format.toxml`(*obj*)

`cterasdk.convert.format.toxmlstr`(*obj, pretty\_print=False*)

Convert a Python object to an XML string

###### Parameters

- **obj** (*object*) – the Python object
- **pretty\_print** (*bool*) – whether to format the XML string, defaults to `False`

**Returns** XML string of the object

**Return type** `str`

##### cterasdk.convert.parse module

`cterasdk.convert.parse.ParseValue`(*data*)

`cterasdk.convert.parse.SetAppendValue`(*item, value*)

`cterasdk.convert.parse.fromjsonstr`(*fromstr*)

`cterasdk.convert.parse.fromxmlstr`(*string*)

## cterasdk.convert.xml\_types module

```

class cterasdk.convert.xml_types.XMLTypes
    Bases: object
    ATT = 'att'
    CLASS = 'class'
    DB = 'db'
    FIRMWARE = 'firmware'
    ID = 'id'
    LIST = 'list'
    LOCATION = 'xsi:noNamespaceSchemaLocation'
    NS = 'xmlns:xsi'
    OBJ = 'obj'
    UUID = 'uuid'
    VAL = 'val'
    VERSION = 'version'

```

## 3.1.4 cterasdk.core package

### 3.1.4.1 Subpackages

#### cterasdk.core.files package

##### Submodules

#### cterasdk.core.files.browser module

```

class cterasdk.core.files.browser.Backups(portal, base_path)
    Bases: cterasdk.core.files.browser.FileBrowser
    Backups File Browser APIs

    device_config(device, destination=None)
        Download a device configuration file

        Parameters
        • device (str) – The device name
        • destination (str, optional) – File destination, if it is a directory, the original filename
          will be kept, defaults to the default directory

class cterasdk.core.files.browser.CloudDrive(portal, base_path)
    Bases: cterasdk.core.files.browser.FileBrowser
    Cloud Drive File Browser APIs

    add_share_recipients(path, recipients)
        Add share recipients

```

**Parameters**

- **path** (*str*) – The path of the file or folder
- **recipients** (*list[cterasdk.core.types.ShareRecipient]*) – A list of share recipients

**Returns** A list of all recipients added

**Return type** *list[cterasdk.core.types.ShareRecipient]*

**delete**(*path*)

Delete a file

**Parameters** **path** (*str*) – Path of the file or directory to delete

**delete\_multi**(*\*args*)

Delete multiple files and/or directories

**Parameters** **\*args** – Variable lengthed list of paths of files and/or directories to delete

**get\_share\_info**(*path*)

Get share settings and recipients

**mkdir**(*path, recurse=False*)

Create a new directory

**Parameters**

- **path** (*str*) – Path of the directory to create
- **recurse** (*bool, optional*) – Whether to create the path recursively, defaults to False

**move**(*src, dest*)

Move a file or directory

**Parameters**

- **src** (*str*) – The source path of the file or directory
- **dst** (*str*) – The destination path of the file or directory

**move\_multi**(*src, dest*)**remove\_share\_recipients**(*path, accounts*)

Remove share recipients

**Parameters**

- **path** (*str*) – The path of the file or folder
- **accounts** (*list[cterasdk.core.types.PortalAccount]*) – A list of portal user or group accounts

**Returns** A list of all share recipients removed

**Return type** *list[cterasdk.core.types.PortalAccount]*

**rename**(*path, name*)

Rename a file

**Parameters**

- **path** (*str*) – Path of the file or directory to rename
- **name** (*str*) – The name to rename to

**share**(*path*, *recipients*, *as\_project=True*, *allow\_reshare=True*, *allow\_sync=True*)

Share a file or a folder

**Parameters**

- **path** (*str*) – The path of the file or folder to share
- **recipients** (*list[cterasdk.core.types.ShareRecipient]*) – A list of share recipients
- **as\_project** (*bool, optional*) – Share as a team project, defaults to True when the item is a cloud folder else False
- **allow\_reshare** (*bool, optional*) – Allow recipients to re-share this item, defaults to True
- **allow\_sync** (*bool, optional*) – Allow recipients to sync this item, defaults to True when the item is a cloud folder else False

**Returns** A list of all recipients added to the collaboration share

**Return type** *list[cterasdk.core.types.ShareRecipient]*

**undelete**(*path*)

Restore a previously deleted file or directory

**Parameters** **path** (*str*) – Path of the file or directory to restore

**undelete\_multi**(*\*args*)

Restore previously deleted multiple files and/or directories

**Parameters** **\*args** – Variable length list of paths of files and/or directories to restore

**unshare**(*path*)

Unshare a file or a folder

**upload**(*file\_path*, *server\_path*)

Upload a file

**Parameters**

- **file\_path** (*str*) – Path to the local file to upload
- **server\_path** (*str*) – Path to the directory to upload the file to

**class** `cterasdk.core.files.browser.FileBrowser`(*portal*, *base\_path*)

Bases: *cterasdk.core.base\_command.BaseCommand*

Portal File Browser APIs

**copy**(*src*, *dest*)

Copy a file or directory

**Parameters**

- **src** (*str*) – The source path of the file or directory
- **dst** (*str*) – The destination path of the file or directory

**copy\_multi**(*src*, *dest*)

**download**(*path*, *destination=None*)

Download a file

**Parameters**

- **path** (*str*) – Path of the file to download

- **destination** (*str, optional*) – File destination, if it is a directory, the original filename will be kept, defaults to the default directory

**download\_as\_zip**(*cloud\_directory, files, destination=None*)

Download a list of files and/or directories from a cloud folder as a ZIP file

**Warning:** The list of files is not validated. The ZIP file will include only the existing files and directories

**Parameters**

- **cloud\_directory** (*str*) – Path to the cloud directory
- **files** (*list[str]*) – List of files and/or directories in the cloud folder to download
- **destination** (*str, optional*) – File destination, if it is a directory, the original filename will be kept, defaults to the default directory

**ls**(*path, include\_deleted=False*)

Execute ls on the provided path

**Parameters**

- **path** (*str*) – Path to list
- **include\_deleted** (*bool, optional*) – Include deleted files, defaults to False

**mlink**(*path, access='RO', expire\_in=30*)

Create a link to a file

**Parameters**

- **path** (*str*) – The path of the file to create a link to
- **access** (*str, optional*) – Access policy of the link, defaults to 'RO'
- **expire\_in** (*int, optional*) – Number of days until the link expires, defaults to 30

**mkpath**(*array*)

**walk**(*path, include\_deleted=False*)

Perform walk on the provided path

**Parameters**

- **path** (*str*) – Path to perform walk on
- **include\_deleted** (*bool, optional*) – Include deleted files, defaults to False

**cterasdk.core.files.collaboration module**

`cterasdk.core.files.collaboration.add_share_recipients(ctera_host, path, recipients)`

`cterasdk.core.files.collaboration.get_share_info(ctera_host, path)`

`cterasdk.core.files.collaboration.remove_share_recipients(ctera_host, path, accounts)`

`cterasdk.core.files.collaboration.share(ctera_host, path, recipients, as_project, allow_reshare, allow_sync)`

`cterasdk.core.files.collaboration.unshare(ctera_host, path)`



**cterasdk.core.files.common module**

**class** cterasdk.core.files.common.**ActionResourcesParam**

Bases: *cterasdk.common.object.Object*

**add**(*param*)

**static instance**()

**class** cterasdk.core.files.common.**CreateShareParam**(*path, access, expire\_on*)

Bases: *cterasdk.common.object.Object*

**static instance**(*path, access, expire\_on*)

**class** cterasdk.core.files.common.**SrcDstParam**(*src, dest=None*)

Bases: *cterasdk.common.object.Object*

**static instance**(*src, dest=None*)

cterasdk.core.files.common.**get\_resource\_info**(*ctera\_host, path*)

**cterasdk.core.files.cp module**

cterasdk.core.files.cp.**copy**(*ctera\_host, src, dest*)

cterasdk.core.files.cp.**copy\_multi**(*ctera\_host, src, dest*)

**cterasdk.core.files.directory module**

**exception** cterasdk.core.files.directory.**InvalidName**(*message=None, instance=None, \*\*kwargs*)

Bases: *cterasdk.exception.CTERAException*

**exception** cterasdk.core.files.directory.**InvalidPath**(*message=None, instance=None, \*\*kwargs*)

Bases: *cterasdk.exception.CTERAException*

**exception** cterasdk.core.files.directory.**ItemExists**(*message=None, instance=None, \*\*kwargs*)

Bases: *cterasdk.exception.CTERAException*

**exception** cterasdk.core.files.directory.**ReservedName**(*message=None, instance=None, \*\*kwargs*)

Bases: *cterasdk.exception.CTERAException*

cterasdk.core.files.directory.**mkdir**(*ctera\_host, path, recurse=False*)

**cterasdk.core.files.file\_access module**

**class** cterasdk.core.files.file\_access.**FileAccess**(*ctera\_host*)

Bases: *cterasdk.lib.file\_access\_base.FileAccessBase*

### **cterasdk.core.files.ln module**

`cterasdk.core.files.ln.mklink`(*ctera\_host, path, access, expire\_in*)

### **cterasdk.core.files.ls module**

`cterasdk.core.files.ls.fetch_resources`(*ctera\_host, param*)

`cterasdk.core.files.ls.list_dir`(*ctera\_host, param*)

`cterasdk.core.files.ls.ls`(*ctera\_host, path, depth=1, include\_deleted=False*)

### **cterasdk.core.files.mv module**

`cterasdk.core.files.mv.move`(*ctera\_host, src, dest*)

`cterasdk.core.files.mv.move_multi`(*ctera\_host, src, dest*)

### **cterasdk.core.files.path module**

**class** `cterasdk.core.files.path.CTERAPath`(*item, basepath*)

Bases: object

`encoded_fullpath`()

`encoded_parent`()

`fullpath`()

`joinpath`(*path*)

`name`()

`parent`()

`parts`()

### **cterasdk.core.files.recover module**

`cterasdk.core.files.recover.undelete`(*ctera\_host, path*)

`cterasdk.core.files.recover.undelete_multi`(*ctera\_host, \*paths*)

### **cterasdk.core.files.rename module**

`cterasdk.core.files.rename.rename`(*ctera\_host, path, name*)

**cterasdk.core.files.rm module**

`cterasdk.core.files.rm.delete(ctera_host, path)`

`cterasdk.core.files.rm.delete_multi(ctera_host, *paths)`

**3.1.4.2 Submodules****cterasdk.core.activation module**

**class** `cterasdk.core.activation.Activation(portal)`

Bases: `cterasdk.core.base_command.BaseCommand`

Portal activation

**generate\_code**(*username=None, tenant=None*)

Generate device activation code

**Parameters**

- **username** (*str, optional*) – User name used for activation, defaults to None
- **tenant** (*str, optional*) – Tenant name used for activation, defaults to None

**Returns** Portal Activation Code

**Return type** str

**cterasdk.core.antivirus module**

**class** `cterasdk.core.antivirus.Antivirus(portal)`

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Antivirus APIs

**Variables** **servers** (`cterasdk.core.antivirus.AntivirusServers`) – Object holding the Portal antivirus server APIs

**default** = ['name', 'type']

**list\_servers**(*include=None*)

List the antivirus servers

**Parameters** **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'type']

**rescan**()

Scan all files using the latest antivirus update. This may take a while

**status**()

Get antivirus service status

**suspend**()

Suspend antivirus scanning

**unsuspend**()

Unsuspend antivirus scanning

**class** `cterasdk.core.antivirus.AntivirusServers(portal)`

Bases: `cterasdk.core.base_command.BaseCommand`

**add**(*name*, *vendor*, *url*, *connection\_timeout=5*)

Add an antivirus server

**Parameters**

- **name** (*str*) – Server name
- **vendor** (`cterasdk.core.enum.AntivirusType`) – Server type
- **url** (*str*) – Server URL (example: `http://your-antivirus.server.local:1234/signature`)
- **connection\_timeout** (*int*, *optional*) – Server connection timeout (in seconds), defaults to 5 seconds

**delete**(*name*)

Remove an antivirus server

**get**(*name*)

Get an antivirus server's configuration

**Parameters** **name** (*str*) – Server name

**suspend**(*name*)

Suspend an antivirus server

**unsuspend**(*name*)

Unsuspend antivirus scanning

### cterasdk.core.base\_command module

**class** `cterasdk.core.base_command.BaseCommand`(*portal*)

Bases: `object`

Base class for all Portal API classes

**session**()

### cterasdk.core.buckets module

**class** `cterasdk.core.buckets.Buckets`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Storage Node APIs

**add**(*name*, *bucket*, *read\_only=False*, *dedicated\_to=None*)

Add a Bucket

**Parameters**

- **name** (*str*) – Name of the bucket
- **bucket** (`cterasdk.core.types.Bucket`) – Storage bucket to add
- **read\_only** (*bool*, *optional*) – Set bucket to read-delete only, defaults to `False`
- **dedicated\_to** (*str*, *optional*) – Name of a tenant, defaults to `None`

**default** = ['name']

**delete**(*name*)

Delete a Bucket

**Parameters** **name** (*str*) – Name of the bucket

**get**(*name*, *include=None*)

Get a Bucket

**Parameters**

- **name** (*str*) – Name of the bucket
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**list\_buckets**(*include=None*)

List Buckets.

To retrieve buckets, you must first browse the Global Administration Portal, using: *GlobalAdmin.portals.browse\_global\_admin()*

**Parameters** **include** (*list[str]*, *optional*) – List of fields to retrieve, defaults to ['name']

**modify**(*current\_name*, *new\_name=None*, *read\_only=None*, *dedicated\_to=None*)

Modify a Bucket

**Parameters**

- **current\_name** (*str*) – The current bucket name
- **new\_name** (*str*, *optional*) – New name
- **read\_only** (*bool*, *optional*) – Set bucket to read-delete only
- **dedicated** (*bool*, *optional*) – Dedicate bucket to a tenant
- **dedicated\_to** (*str*, *optional*) – Tenant name

**read\_only**(*name*)

Set bucket to Read Only

**Parameters** **name** (*str*) – Name of the bucket

**read\_write**(*name*)

Set bucket to Read Write

**Parameters** **name** (*str*) – Name of the bucket

### cterasdk.core.cloudfs module

**class** `cterasdk.core.cloudfs.CloudFS`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

CloudFS APIs

**default** = ['name', 'group', 'owner']

**delete**(*name*, *owner*)

Delete a Cloud Drive Folder

**Parameters**

- **name** (*str*) – Name of the Cloud Drive Folder to delete
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the Cloud Drive Folder to delete

**find**(*name*, *owner*, *include=None*)

Find a Cloud Drive Folder

**Parameters**

- **name** (*str*) – Name of the Cloud Drive Folder to find
- **owner** (`cterasdk.core.types.UserAccount`) – User account of the folder group owner
- **include** (*list[str]*) – List of metadata fields to include in the response

**Returns** A Cloud Drive Folder

**get**(*name, include=None*)

Get folder group

**Parameters**

- **name** (*str*) – Name of the Folder Group to find
- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'owner']

**list\_folder\_groups**(*include=None, user=None*)

List folder groups

**Parameters**

- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'owner']
- **user** (`cterasdk.core.types.UserAccount`) – User account of the folder group owner

**Returns** Iterator for all folder groups

**list\_folders**(*include=None, list\_filter='NonDeleted', user=None*)

List Cloud Drive folders.

**Parameters**

- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'group', 'owner']
- **list\_filter** (`cterasdk.core.enum.ListFilter`) – Filter the list of Cloud Drive folders, defaults to non-deleted folders
- **user** (`cterasdk.core.types.UserAccount`) – User account of the cloud folder owner

**Returns** Iterator for all Cloud Drive folders

**mkdir**(*name, group, owner, winacls=True, description=None*)

Create a new directory

**Parameters**

- **name** (*str*) – Name of the new directory
- **group** (*str*) – The Folder Group to which the directory belongs
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the new directory
- **winacls** (*bool, optional*) – Use Windows ACLs, defaults to True
- **description** (*str, optional*) – Cloud drive folder description

**mkfg**(*name, user=None, deduplication\_method\_type=None*)

Create a new Folder Group

**Parameters**

- **name** (*str*) – Name of the new folder group
- **user** (`cterasdk.core.types.UserAccount`) – User account, the user directory and name of the new folder group owner (default to None)

- **deduplication\_method\_type** (`cterasdk.core.enum.DeduplicationMethodType`) – Deduplication-Method

**rmfg**(*name*)

Remove a Folder Group

**Parameters** **name** (*str*) – Name of the folder group to remove

**undelete**(*name, owner*)

Un-Delete a Cloud Drive Folder

**Parameters**

- **name** (*str*) – Name of the Cloud Drive Folder to un-delete
- **owner** (`cterasdk.core.types.UserAccount`) – User account, the owner of the Cloud Drive Folder to delete

### cterasdk.core.connection module

`cterasdk.core.connection.test(CTERAHost)`

`cterasdk.core.connection.test_network(CTERAHost)`

### cterasdk.core.decorator module

`cterasdk.core.decorator.update_current_tenant(function)`

### cterasdk.core.devices module

**class** `cterasdk.core.devices.Devices(portal)`

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Devices APIs

**agents**(*include=None, allPortals=False*)

Get Agents

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False

**Returns** Iterator for all matching Agents

**Return type** `cterasdk.lib.iterator.Iterator[cterasdk.object.Agent.Agent]`

**by\_name**(*names, include=None*)

Get Devices by their names

**Parameters**

- **names** (*list[str], optional*) – List of names of devices
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']

**Returns** Iterator for all matching Devices

**Return type** *cterasdk.lib.iterator.Iterator*

**default** = ['name', 'portal', 'deviceType']

**desktops**(*include=None, allPortals=False*)

Get Desktops

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False

**Returns** Iterator for all matching Desktops

**Return type** *cterasdk.lib.iterator.Iterator*

**device**(*device\_name, tenant=None, include=None*)

Get a Device by its name

**Parameters**

- **device\_name** (*str*) – Name of the device to retrieve
- **tenant** (*str, optional*) – Tenant of the device, defaults to the tenant in the current session
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']

**Returns** Managed Device

**Return type** *ctera.object.Gateway.Gateway* or *ctera.object.Agent.Agent*

**devices**(*include=None, allPortals=False, filters=None, user=None*)

Get Devices

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False
- **filters** (*list[], optional*) – List of additional filters, defaults to None
- **user** (*cterasdk.core.types.UserAccount*) – User account of the device owner

**Returns** Iterator for all matching Devices

**Return type** *cterasdk.lib.iterator.Iterator*

**filers**(*include=None, allPortals=False, deviceTypes=None*)

Get Filers

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False
- **deviceTypes** (*list[cterasdk.core.enum.DeviceType.Gateways]*) – Types of Filers, defaults to all Filer types

**Returns** Iterator for all matching Filers



**Return type** `cterasdk.lib.iterator.Iterator[cterasdk.object.Gateway.Gateway]`

**name\_attr** = 'name'

**servers** (*include=None, allPortals=False*)

Get Servers

#### Parameters

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name', 'portal', 'deviceType']
- **allPortals** (*bool, optional*) – Search in all portals, defaults to False

**Returns** Iterator for all matching Servers

**Return type** `cterasdk.lib.iterator.Iterator`

**type\_attr** = 'deviceType'

### cterasdk.core.directoryservice module

**class** `cterasdk.core.directoryservice.DirectoryService` (*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Active Directory APIs

**connect** (*domain, username, password, directory='ActiveDirectory', domain\_controllers=None, ou=None, ssl=False, krb=False, mapping=None, acl=None, default='Disabled', fetch='Lazy'*)

Connect a Portal tenant to directory services

#### Parameters

- **domain** (*str*) – The directory services domain to connect to
- **username** (*str*) – The user name to use when connecting to the active directory services
- **password** (*str*) – The password to use when connecting to the active directory services
- **ou** (*str, optional*) – The OU path to use when connecting to the active directory services, defaults to *None*
- **directory** (`cterasdk.core.enum.DirectoryServiceType`, *optional*) – The directory service type, defaults to 'ActiveDirectory'
- **domain\_controllers** (`cterasdk.core.types.DomainControllers`, *optional*) – Connect to a primary and a secondary domain controllers, defaults to *None*
- **ssl** (*bool, optional*) – Connect using SSL, defaults to *False*
- **krb** (*bool, optional*) – Connect using Kerberos, defaults to *False*
- **list[cterasdk.core.types.ADDomainIDMapping], optional** – The directory services UID/GID mapping
- **acl** (*list[cterasdk.core.types.AccessControlEntry], optional*) – List of access control entries and their associated roles
- **default** (`cterasdk.core.enum.Role`) – Default role if no match applies, defaults to *None*
- **fetch** (*str, optional*) – Configure identity fetching mode, defaults to 'Lazy'

**connected**()

**disconnect()**

Disconnect a Portal tenant from directory services

**domains()**

Get domains

**Return list(str)** List of names of all discovered domains

**fetch(*active\_directory\_accounts*)**

Instruct the Portal to fetch the provided Active Directory Accounts

**Parameters** **active\_directory\_accounts** (*list[cterasdk.core.types.PortalAccount]*) – List of Active Directory Accounts to fetch

**Returns** Response Code

**get\_access\_control()**

Retrieve directory services access control list

**Returns** List of access control entries

**Return type** *list[cterasdk.core.types.AccessControlEntry]*

**get\_advanced\_mapping()**

Retrieve directory services advanced mapping configuration

**Returns** A dictionary of domain mapping objects

**Return type** dict

**get\_connected\_domain()**

Get the connected domain information. Returns *None* if the Portal tenant is not connected to a domain

**Return str** The connected domain

**get\_default\_role()**

Retrieve the default role assigned when no access control entry match was found

**set\_access\_control(*acl=None, default=None*)**

Configure directory services access control

**Parameters**

- **acl** (*list[cterasdk.core.types.AccessControlEntry], optional*) – List of access control entries and their associated roles
- **default** (*cterasdk.core.enum.Role*) – Default role if no match applies, defaults to *None*

**set\_advanced\_mapping(*mapping*)**

Configure advanced mapping

**Parameters** **mapping** (*list[cterasdk.core.types.ADDomainIDMapping]*) – The directory services UID/GID mapping

**cterasdk.core.enum module****class** cterasdk.core.enum.**AntivirusType**

Bases: object

Antivirus Type

**Variables**

- **McAfeeWG** (*str*) – McAfee Web Gateway
- **Symantec** (*str*) – Symantec Protection Engine
- **ESET** (*str*) – ESET Gateway Security
- **Sophos** (*str*) – Sophos AV
- **McAfeeVSES** (*str*) – McAfee VirusScan Enterprise for Storage
- **TrendMicro** (*str*) – Trend Micro InterScan

**ESET** = 'Eset'**McAfeeVSES** = 'McAfeeVSES'**McAfeeWG** = 'McAfee'**Sophos** = 'Sophos'**Symantec** = 'Symantec'**TrendMicro** = 'TrendMicro'**class** cterasdk.core.enum.**BucketType**

Bases: object

Bucket Type

**Variables**

- **Azure** (*str*) – Azure
- **Scality** (*str*) – Scality
- **AWS** (*str*) – Amazon Web Services S3
- **ICOS** (*str*) – IBM Cloud Object Storage
- **S3Compatible** (*str*) – Generic S3
- **Nutanix** (*str*) – Nutanix S3
- **Wasabi** (*str*) – Wasabi S3
- **Google** (*str*) – Google S3
- **NetAppStorageGRID** (*str*) – NetApp StorageGRID WebScale (S3)

**AWS** = 'S3'**Azure** = 'Azure'**Google** = 'GoogleS3'**ICOS** = 'CleverSafeS3'**NetAppStorageGRID** = 'NTAP'**Nutanix** = 'Nutanix'

**S3Compatible** = 'GenericS3'

**Scality** = 'ScalityS3'

**Wasabi** = 'WasabiS3'

**class** cterasdk.core.enum.CollaboratorType

Bases: object

Collaborator Type

**Variables**

- **LU** (*str*) – Local User
- **DU** (*str*) – Domain User
- **LG** (*str*) – Local Group
- **DG** (*str*) – Domain Group
- **EXT** (*str*) – External

**DG** = 'adGroup'

**DU** = 'adUser'

**EXT** = 'external'

**LG** = 'localGroup'

**LU** = 'localUser'

**class** cterasdk.core.enum.Context

Bases: object

Portal connection context

**Variables**

- **admin** (*str*) – Global admin context
- **ServicesPortal** (*str*) – Services Portal context

**ServicesPortal** = 'ServicesPortal'

**admin** = 'admin'

**class** cterasdk.core.enum.DeduplicationMethodType

Bases: object

Folder Group Deduplication Method Type

**Variables**

- **AverageBlockSize** (*str*) – AverageBlockSize
- **FixedBlockSize** (*str*) – FixedBlockSize

**AverageBlockSize** = 'AverageBlockSize'

**FixedBlockSize** = 'FixedBlockSize'

**class** cterasdk.core.enum.DeviceType

Bases: object

Device type

**Variables**

- **CloudPlug** (*str*) – Cloud Plug device
- **C200** (*str*) – C200 device
- **C400** (*str*) – C400 device
- **C800** (*str*) – C800 device
- **C800P** (*str*) – C800P device
- **vGateway** (*str*) – vGateway device
- **ServerAgent** (*str*) – Server Agent device
- **WorkstationAgent** (*str*) – Workstation Agent Agent device
- **Gateways** (*list[str]*) – List of all the Gateway DeviceTypes
- **Agents** (*list[str]*) – List of all the Agents DeviceTypes

```
Agents = ['Server Agent', 'Workstation Agent']
```

```
C200 = 'C200'
```

```
C400 = 'C400'
```

```
C800 = 'C800'
```

```
C800P = 'C800P'
```

```
CloudPlug = 'CloudPlug'
```

```
Gateways = ['CloudPlug', 'C200', 'C400', 'C800', 'C800P', 'vGateway']
```

```
ServerAgent = 'Server Agent'
```

```
WorkstationAgent = 'Workstation Agent'
```

```
vGateway = 'vGateway'
```

```
class cterasdk.core.enum.DirectorySearchEntityType
```

```
Bases: object
```

```
Directory Search Entity Type
```

#### Variables

- **User** (*str*) – User
- **Group** (*str*) – Group

```
Group = 'group'
```

```
User = 'user'
```

```
class cterasdk.core.enum.DirectoryServiceFetchMode
```

```
Bases: object
```

```
Directory Service Fetch Mode
```

#### Variables

- **Eager** (*str*) – Eager
- **Lazy** (*str*) – Lazy

```
Eager = 'Eager'
```

```
Lazy = 'Lazy'
```

```
class cterasdk.core.enum.DirectoryServiceType
```

```
Bases: object
```

```
Directory Service Type
```

```
Variables
```

- **Microsoft** (*str*) – Active Directory
- **LDAP** (*str*) – LDAP
- **Apple** (*str*) – Apple Open Directory

```
Apple = 'AppleOpenDirectory'
```

```
LDAP = 'LDAP'
```

```
Microsoft = 'ActiveDirectory'
```

```
class cterasdk.core.enum.EnvironmentVariables
```

```
Bases: object
```

```
Environment Variables.
```

Some environment variables are applicable across platforms (i.e. Windows, Linux), while others are limited to a designated platform

```
Variables
```

- **ALLUSERSPROFILE** (*str*) – All users profile
- **WINDIR** (*str*) – Windows directory
- **TEMP** (*str*) – Temp directory
- **SYSTEMDRIVE** (*str*) – System drive
- **PROGRAMFILES** (*str*) – Program files
- **APPDATA** (*str*) – Application data
- **USERPROFILE** (*str*) – Current user profile
- **PRIMARYUSER** (*str*) – Primary user
- **USERS** (*str*) – Users directory (CTERA Edge Filer)
- **AGENTS** (*str*) – Agents directory (CTERA Edge Filer)
- **SYNCS** (*str*) – Syncs directory (CTERA Edge Filer)
- **PROJECTS** (*str*) – Projects directory (CTERA Edge Filer)

```
AGENTS = '$AGENTS'
```

```
ALLUSERSPROFILE = '$ALLUSERSPROFILE'
```

```
APPDATA = '$APPDATA'
```

```
PRIMARYUSER = '$PRIMARYUSER'
```

```
PROGRAMFILES = '$PROGRAMFILES'
```

```
PROJECTS = '$PROJECTS'
```

```
SYNCS = '$SYNCS'
```

```
SYSTEMDRIVE = '$SYSTEMDRIVE'
```

```
TEMP = '$TEMP'
```

```
USERPROFILE = '$USERPROFILE'
```

```
USERS = '$USERS'
```

```
WINDIR = '$WINDIR'
```

```
class cterasdk.core.enum.FileAccessMode
```

```
Bases: object
```

```
Share Access Mode
```

#### Variables

- **RW** (*str*) – Read Write
- **RO** (*str*) – Read Only
- **PO** (*str*) – Preview Only
- **NA** (*str*) – None

```
NA = 'None'
```

```
PO = 'PreviewOnly'
```

```
RO = 'ReadOnly'
```

```
RW = 'ReadWrite'
```

```
class cterasdk.core.enum.ICAPServices
```

```
Bases: object
```

```
ICAP Services
```

#### Variables

- **Antivirus** (*str*) – Antivirus
- **DLP** (*str*) – Data Loss Prevention

```
Antivirus = 'Antivirus'
```

```
DLP = 'DLP'
```

```
class cterasdk.core.enum.IPProtocol
```

```
Bases: object
```

```
IP Protocol
```

#### Variables

- **TCP** (*str*) – TCP Protocol
- **UDP** (*str*) – UDP Protocol

```
TCP = 'TCP'
```

```
UDP = 'UDP'
```

```
class cterasdk.core.enum.ListFilter
```

```
Bases: object
```

```
Cloud Drive Folder List Filter
```

#### Variables

- **All** (*str*) – All
- **Deleted** (*str*) – Deleted

- **NonDeleted** (*str*) – NonDeleted

**All** = 'All'

**Deleted** = 'Deleted'

**NonDeleted** = 'NonDeleted'

**class** cterasdk.core.enum.**LocationType**

Bases: object

Location Type

**Variables**

- **Azure** (*str*) – Azure Blob Storage
- **S3** (*str*) – Amazon Web Services S3
- **S3Compatible** (*str*) – S3 Compatible
- **NetAppStorageGRID** (*str*) – NetApp StorageGRID WebScale (S3)

**Azure** = 'AzureLocation'

**NetAppStorageGRID** = 'NetAppLocation'

**S3** = 'S3Location'

**S3Compatible** = 'S3Compatible'

**class** cterasdk.core.enum.**LogTopic**

Bases: object

Portal Log Topic

**Variables**

- **System** (*str*) – System log topic
- **CloudBackup** (*str*) – Cloud Backup log topic
- **CloudSync** (*str*) – Cloud Sync log topic
- **Access** (*str*) – Access log topic
- **Audit** (*str*) – Audit log topic

**Access** = 'access'

**Audit** = 'audit'

**CloudBackup** = 'backup'

**CloudSync** = 'cloudsync'

**System** = 'system'

**class** cterasdk.core.enum.**Mode**

Bases: object

Enum for operational mode

**Variables**

- **Enabled** (*str*) – Operational mode enabled
- **Disabled** (*str*) – Operational mode disabled

**Disabled** = 'disabled'



**Enabled = 'enabled'**

**class** cterasdk.core.enum.OriginType

Bases: object

Log Origin Type

**Variables**

- **Portal** (*str*) – Portal originated logs
- **Device** (*str*) – Device originated logs

**Device = 'Device'**

**Portal = 'Portal'**

**class** cterasdk.core.enum.PlanCriteria

Bases: object

Subscription Plan Auto Assignment Rule Builder Criterias

**Variables**

- **Username** (*str*) – Username
- **Groups** (*str*) – User groups
- **Role** (*str*) – User role
- **First** (*str*) – User first name
- **Last** (*str*) – User last name
- **Company** (*str*) – User company
- **BillingId** (*str*) – User billing id
- **Comment** (*str*) – User comment

**BillingId = 'billingId'**

**Comment = 'comment'**

**Company = 'company'**

**First = 'firstName'**

**Groups = 'userGroups'**

**Last = 'lastName'**

**Role = 'role'**

**Username = 'username'**

**class** cterasdk.core.enum.PlanItem

Bases: object

Portal plan item

**Variables**

- **EV4** (*str*) – EV4
- **EV8** (*str*) – EV8
- **EV16** (*str*) – EV16
- **EV32** (*str*) – EV32

- **EV64** (*str*) – EV64
- **EV128** (*str*) – EV128
- **WA** (*str*) – Workstation Agent
- **SA** (*str*) – Server Agent
- **Share** (*str*) – Cloud Drive
- **Connect** (*str*) – Cloud Drive Connect

**Connect** = 'Connect'

**EV128** = 'EV128'

**EV16** = 'EV16'

**EV32** = 'EV32'

**EV4** = 'EV4'

**EV64** = 'EV64'

**EV8** = 'EV8'

**SA** = 'SA'

**Share** = 'Share'

**Storage** = 'Storage'

**WA** = 'WA'

**class** cterasdk.core.enum.**PlanRetention**

Bases: object

Portal plan retention policy

### Variables

- **All** (*str*) – All versions
- **Hourly** (*str*) – Hourly versions
- **Daily** (*str*) – Daily versions
- **Weekly** (*str*) – Weekly versions
- **Monthly** (*str*) – Monthly versions
- **Quarterly** (*str*) – Quarterly versions
- **Yearly** (*str*) – Yearly versions
- **Deleted** (*str*) – Recycle bin

**All** = 'retainAll'

**Daily** = 'daily'

**Deleted** = 'retainDeleted'

**Hourly** = 'hourly'

**Monthly** = 'monthly'

**Quarterly** = 'quarterly'

**Weekly** = 'weekly'

**Yearly** = 'yearly'

**class** cterasdk.core.enum.**Platform**

Bases: object

CTERA Edge Platform Type.

**Variables**

- **C200\_Orion** (*str*) – All users profile
- **C200\_ARM** (*str*) – Windows directory
- **C200\_Kirkwood** (*str*) – Temp directory
- **C400\_C800** (*str*) – System drive
- **Edge\_6** (*str*) – CTERA 6.0 Edge Filer
- **Edge\_7** (*str*) – CTERA 7.0 Edge Filer
- **Windows** (*str*) – Windows Agent (Drive App)
- **Linux** (*str*) – Linux Agent (Drive App)
- **OSX** (*str*) – Mac Agent (Drive App)

**C200\_ARM** = 'ARM'

**C200\_Kirkwood** = 'Kirkwood'

**C200\_Orion** = 'Orion'

**C400\_C800** = 'X86'

**Edge\_6** = 'VBox'

**Edge\_7** = 'Genesis'

**Linux** = 'LinuxX86'

**OSX** = 'OSxX86'

**Windows** = 'WindowsX86'

**class** cterasdk.core.enum.**PolicyType**

Bases: object

Zone Policy Type

**Variables**

- **ALL** (*str*) – All folders
- **SELECT** (*str*) – Selected Folders
- **NONE** (*str*) – No Folders

**ALL** = 'allFolders'

**NONE** = 'noFolders'

**SELECT** = 'selectedFolders'

**class** cterasdk.core.enum.**PortalAccountType**

Bases: object

Portal Account Type

**Variables**

- **User** (*str*) – User account type
- **Group** (*str*) – Group account type

**Group** = 'group'

**User** = 'user'

**class** cterasdk.core.enum.**PortalType**

Bases: object

Portal Type

### Variables

- **Team** (*str*) – Team Portal
- **Reseller** (*str*) – Reseller Portal

**Reseller** = 'reseller'

**Team** = 'team'

**class** cterasdk.core.enum.**ProtectionLevel**

Bases: object

External Share Protection Level

### Variables

- **publicLink** (*str*) – No authentication
- **email** (*str*) – 2FA via email

**Email** = 'email'

**Public** = 'publicLink'

**class** cterasdk.core.enum.**Role**

Bases: object

Portal User Role

### Variables

- **Disabled** (*str*) – Disabled user role
- **EndUser** (*str*) – EndUser user role
- **ReadWriteAdmin** (*str*) – ReadWriteAdmin user role
- **ReadOnlyAdmin** (*str*) – ReadOnlyAdmin user role
- **Support** (*str*) – Support user role

**Disabled** = 'Disabled'

**EndUser** = 'EndUser'

**ReadOnlyAdmin** = 'ReadOnlyAdmin'

**ReadWriteAdmin** = 'ReadWriteAdmin'

**Support** = 'Support'

**class** cterasdk.core.enum.**SearchType**

Bases: object

Search Type

**Variables**

- **User** (*str*) – User search type
- **Group** (*str*) – Group search type

**Groups** = 'groups'**Users** = 'users'**class** cterasdk.core.enum.**ServerMode**

Bases: object

Portal Server Mode

**Variables**

- **Master** (*str*) – Master
- **Slave** (*str*) – Slave

**Master** = 'master'**Slave** = 'slave'**class** cterasdk.core.enum.**SetupWizardStage**

Bases: object

Portal Setup Wizard Stage

**Variables**

- **Server** (*str*) – Initializing Server
- **Portal** (*str*) – Initializing Portal
- **Replication** (*str*) – Setting Database Replication
- **Restart** (*str*) – Restarting Server
- **Finish** (*str*) – Finished

**Finish** = 'finish'**Portal** = 'initPortal'**Replication** = 'setReplication'**Restart** = 'restartingServer'**Server** = 'initServer'**class** cterasdk.core.enum.**SetupWizardStatus**

Bases: object

Portal Setup Wizard Status

**Variables**

- **NA** (*str*) – Not Relevant
- **Running** (*str*) – In Progress
- **Failed** (*str*) – Failed
- **Completed** (*str*) – Completed

**Completed** = 'completed'**Failed** = 'failed'

**NA** = 'notRelevant'

**Running** = 'InProgress'

**class** cterasdk.core.enum.**Severity**

Bases: object

Portal Log Severity

**Variables**

- **EMERGENCY** (*str*) – Emergency log severity
- **ALERT** (*str*) – Alert log severity
- **CRITICAL** (*str*) – Critical log severity
- **ERROR** (*str*) – Error log severity
- **WARNING** (*str*) – Warning log severity
- **NOTICE** (*str*) – Notice log severity
- **INFO** (*str*) – Info log severity
- **DEBUG** (*str*) – Debug log severity

**ALERT** = 'alert'

**CRITICAL** = 'critical'

**DEBUG** = 'debug'

**EMERGENCY** = 'emergency'

**ERROR** = 'error'

**INFO** = 'info'

**NOTICE** = 'notice'

**WARNING** = 'warning'

**class** cterasdk.core.enum.**SlaveAuthenticaiionMethod**

Bases: object

Secondary Portal server authentication mode

**Variables**

- **Password** (*str*) – Password
- **PrivateKey** (*str*) – Private Key

**Password** = 'Password'

**PrivateKey** = 'Key'

**class** cterasdk.core.enum.**TemplateCriteria**

Bases: object

Configuration Template Auto Assignment Rule Builder Criterias

**Variables**

- **Type** (*str*) – Device type
- **OperatingSystem** (*str*) – Operating system
- **Version** (*str*) – Installed software version

- **Hostname** (*str*) – Hostname
- **Name** (*str*) – Device name
- **Owner** (*str*) – Device owner username
- **Plan** (*str*) – Plan name
- **Groups** (*str*) – Device owner local or domain groups

```

Groups = 'ownerGroups'
Hostname = 'Hostname'
Name = 'DeviceName'
OperatingSystem = 'OperatingSystem'
Owner = 'OwnerUsername'
Plan = 'Plan'
Type = 'DeviceType'
Version = 'InstalledSoftwareVersion'

```

### cterasdk.core.login module

```

class cterasdk.core.login.Login(portal)
    Bases: cterasdk.core.base_command.BaseCommand
    Portal Login APIs
    login(username, password)
        Log into the portal
        Parameters
            • username (str) – User name to log in
            • password (str) – User password
    logout()
        Log out of the portal

```

### cterasdk.core.logs module

```

class cterasdk.core.logs.Logs(portal)
    Bases: cterasdk.core.base_command.BaseCommand
    Portal Logs APIs
    device(name, topic='system', min_severity='info', before=None, after=None, filters=None)
        Get device logs from the Portal
        Parameters
            • name (str) – Name of a device
            • topic (cterasdk.core.enum.LogTopic, optional) – Log topic to get, defaults to
              cterasdk.core.enum.LogTopic.System
            • min_severity (cterasdk.core.enum.Severity, optional) – Minimum severity of
              logs to get, defaults to cterasdk.core.enum.Severity.INFO

```

- **before** (*str, optional*) – Get logs before this date (in format “%m/%d/%Y %H:%M:%S”), defaults to None
- **after** (*str, optional*) – Get logs after this date (in format “%m/%d/%Y %H:%M:%S”), defaults to None
- **filters** (*list[cterasdk.core.query.FilterBuilder], optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching logs

**Return type** *cterasdk.lib.iterator.Iterator*[*cterasdk.object.Object*]

**get**(*topic='system', min\_severity='info', origin\_type='Portal', origin=None, before=None, after=None, filters=None*)

Get logs from the Portal

#### Parameters

- **topic** (*cterasdk.core.enum.LogTopic, optional*) – Log topic to get, defaults to *cterasdk.core.enum.LogTopic.System*
- **min\_severity** (*cterasdk.core.enum.Severity, optional*) – Minimum severity of logs to get, defaults to *cterasdk.core.enum.Severity.INFO*
- **origin\_type** (*cterasdk.core.enum.OriginType, optional*) – Origin type of the logs to get, defaults to *cterasdk.core.enum.OriginType.Portal*
- **origin** (*str, optional*) – Log origin (e.g. device name, Portal server name), defaults to None
- **before** (*str, optional*) – Get logs before this date (in format “%m/%d/%Y %H:%M:%S”), defaults to None
- **after** (*str, optional*) – Get logs after this date (in format “%m/%d/%Y %H:%M:%S”), defaults to None
- **filters** (*list[cterasdk.core.query.FilterBuilder], optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching logs

**Return type** *cterasdk.lib.iterator.Iterator*[*cterasdk.object.Object*]

## cterasdk.core.portals module

**class** *cterasdk.core.portals.Portals*(*portal*)

Bases: *cterasdk.core.base\_command.BaseCommand*

Global Admin Portals APIs

**add**(*name, display\_name=None, billing\_id=None, company=None, plan=None, comment=None*)

Add a new tenant

#### Parameters

- **name** (*str*) – Name of the new tenant
- **display\_name** (*str, optional*) – Display Name of the new tenant, defaults to None
- **billing\_id** (*str, optional*) – Billing ID of the new tenant, defaults to None
- **company** (*str, optional*) – Company Name of the new tenant, defaults to None



- **plan** (*str, optional*) – Subscription plan name to assign to the new tenant, defaults to None
- **comment** (*str, optional*) – Assign a comment to the new tenant, defaults to None

**Return str** A relative url path to the Team Portal

**apply\_changes**(*wait=False*)  
Apply provisioning changes.

**Parameters wait** (*bool, optional*) – Wait for all changes to apply

**browse**(*tenant*)  
Browse a tenant

**Parameters tenant** (*str*) – Name of the tenant to browse

**browse\_global\_admin**()  
Browse the Global Admin

**default** = ['name']

**delete**(*name*)  
Delete an existing tenant

**Parameters name** (*str*) – Name of the tenant to delete

**get**(*name, include=None*)  
Get a tenant

**Parameters**

- **name** (*str*) – Name of the tenant
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**list\_tenants**(*include=None, portal\_type=None*)  
List tenants.

To retrieve tenants, you must first browse the Global Administration Portal, using: *GlobalAdmin.portals.browse\_global\_admin()*

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **portal\_type** (*cterasdk.core.enum.PortalType*) – The Portal type

**subscribe**(*tenant, plan*)  
Subscribe a tenant to a plan

**Parameters**

- **name** (*str*) – Name of the tenant
- **str, plan** – Name of the subscription plan

**tenants**(*include\_deleted=False*)  
Get all tenants

**Parameters include\_deleted** (*bool, optional*) – Include deleted tenants, defaults to False

**undelete**(*name*)  
Undelete a previously deleted tenant

**Parameters name** (*str*) – Name of the tenant to undelete

**cterasdk.core.query module**

```
class cterasdk.core.query.Filter(field)
    Bases: cterasdk.common.object.Object

class cterasdk.core.query.FilterBuilder(name, reference=False)
    Bases: cterasdk.common.object.Object

    after(value)
    before(value)
    eq(value)
    ge(value)
    gt(value)
    le(value)
    like(value)
    lt(value)
    ne(value)
    notLike(value)
    static ref(name)
    setValue(value)

class cterasdk.core.query.FilterType
    Bases: object

    Boolean = 'BooleanFilter'
    BooleanRefFilter = 'BooleanRefFilter'
    DateTime = 'DateTimeFilter'
    IntRefFilter = 'IntRefFilter'
    Integer = 'IntFilter'
    RefFilter = 'RefFilter'
    String = 'StringFilter'
    static fromValue(value, ref)

class cterasdk.core.query.QueryParamBuilder
    Bases: object

    addFilter(query_filter)
    allPortals(allPortals)
    build()
    countLimit(countLimit)
    include(include)
    include_classname()
    orFilter(orFilter)
    ownedBy(ownedBy)
```

**put**(*key, value*)

**sortBy**(*sortBy*)

**startFrom**(*startFrom*)

```
class cterasdk.core.query.QueryParams
    Bases: cterasdk.common.object.Object
    include_classname()
    increment()
```

```
class cterasdk.core.query.Restriction
    Bases: object
    EQUALS = 'eq'
    GREATER_EQUALS = 'ge'
    GREATER_THAN = 'gt'
    LESS_EQUALS = 'le'
    LESS_THAN = 'lt'
    LIKE = 'like'
    NOT_EQUALS = 'ne'
    UNLIKE = 'notLike'
```

`cterasdk.core.query.iterator(CTERAHost, path, param)`

`cterasdk.core.query.query(CTERAHost, path, param)`

`cterasdk.core.query.show(CTERAHost, path, param)`

### cterasdk.core.reports module

```
class cterasdk.core.reports.Reports(portal)
    Bases: cterasdk.core.base_command.BaseCommand
    Reports APIs
```

**folder\_groups**()  
Retrieve the folder groups statistics report.

To retrieve this report, you must first browse the Virtual Portal that contains the report, using: *GlobalAdmin.portals.browse()*

**folders**()  
Retrieve the cloud folders statistics report.

To retrieve this report, you must first browse the Virtual Portal that contains the report, using: *GlobalAdmin.portals.browse()*

**portals**()  
Retrieve the storage statistics report.

To retrieve this report, you must first browse the Global Administration Portal, using: *GlobalAdmin.portals.browse\_global\_admin()*

**storage()**

Retrieve the portals statistics report.

To retrieve this report, you must first browse the Global Administration Portal, using: `GlobalAdmin.portals.browse_global_admin()`

**cterasdk.core.plans module**

**class** `cterasdk.core.plans.PlanAutoAssignPolicy`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

**get\_policy()**

Get plans auto assignment policy

**set\_policy**(*rules*, *apply\_default=None*, *default=None*, *apply\_changes=True*)

Set plans auto assignment policy

**Parameters**

- **rules** (*list*[`cterasdk.common.types.PolicyRule`]) – List of policy rules
- **apply\_default** (*bool*, *optional*) – If no match found, apply default plan. If not passed, the current config will be kept
- **default** (*str*, *optional*) – Name of a plan to assign if no match found. Ignored unless the `apply_default` is set to `True`
- **apply\_changes** (*bool*, *optional*) – Apply provisioning changes upon update, defaults to `True`

**class** `cterasdk.core.plans.Plans`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Plan APIs

**Variables** `auto_assign` (`cterasdk.core.plans.PlanAutoAssignPolicy`) – Object holding the Portal subscription plan auto assignment rules APIs

**add**(*name*, *retention=None*, *quotas=None*)

Add a subscription plan

**Parameters**

- **retention** (*dict*, *optional*) – The data retention policy
- **quotas** (*dict*, *optional*) – The items included in the plan and their respective quota

**by\_name**(*names*, *include=None*)

Get Plans by their names

**Parameters**

- **names** (*list*[*str*], *optional*) – List of names of plans
- **include** (*list*[*str*], *optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list*[`cterasdk.core.query.FilterBuilder`], *optional*) – List of additional filters, defaults to `None`

**Returns** Iterator for all matching Plans

**Return type** `cterasdk.lib.iterator.Iterator`

**default** = ['name']

**delete**(*name*)

Delete a subscription plan

**Parameters** **username** (*str*) – The name of the subscription plan

**get**(*name*, *include=None*)

Retrieve subscription plan properties

**Parameters**

- **name** (*str*) – Name of the subscription plan
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** The subscription plan, including the requested fields

**list\_plans**(*include=None*, *filters=None*)

List Plans

**Parameters**

- **include** (*list[str]*, *optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[]*, *optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching Plans

**Return type** *cterasdk.lib.iterator.Iterator*

**modify**(*name*, *retention=None*, *quotas=None*, *apply\_changes=True*)

Modify a subscription plan

**Parameters**

- **retention** (*dict*, *optional*) – The data retention policy
- **quotas** (*dict*, *optional*) – The items included in the plan and their respective quota
- **apply\_changes** (*bool*, *optional*) – Apply provisioning changes immediately

### cterasdk.core.remote module

cterasdk.core.remote.**remote\_command**(*Portal*, *device*)

### cterasdk.core.servers module

**class** cterasdk.core.servers.**Servers**(*portal*)

Bases: *cterasdk.core.base\_command.BaseCommand*

Global Admin Servers APIs

**default** = ['name']

**get**(*name*, *include=None*)

Retrieve server properties

**Parameters**

- **name** (*str*) – Name of the server
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** The server, including the requested fields

**list\_servers**(*include=None*)

Retrieve the servers that comprise CTERA Portal.

To retrieve servers, you must first browse the Global Administration Portal, using: `GlobalAdmin.portals.browse_global_admin()`

**Parameters include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']

**modify**(*name, server\_name=None, app=None, preview=None, enable\_public\_ip=None, public\_ip=None, allow\_user\_login=None, enable\_replication=None, replica\_of=None*)

Modify a Portal server

**Parameters**

- **name** (*str*) – The current server name
- **server\_name** (*str, optional*) – New server name
- **app** (*bool, optional*) – Application server
- **preview** (*bool, optional*) – Preview server
- **enable\_public\_ip** (*bool, optional*) – Enable or disable public NAT address
- **public\_ip** (*str, optional*) – Public NAT address
- **allow\_user\_login** (*bool, optional*) – Allow or disallow logins to this server
- **enable\_replication** (*bool, optional*) – Enable or disable database replication
- **replica\_of** (*str, optional*) – Configure as a replicate of another Portal server. *enable\_replication* must be set to *True*

### cterasdk.core.session module

**class** `cterasdk.core.session.Session`(*host, context*)

Bases: `cterasdk.lib.session_base.SessionBase`

**Administration** = 'Administration'

**in\_tenant\_context**()

**is\_global\_admin**()

**update\_tenant**(*current\_tenant*)

### cterasdk.core.setup module

**class** `cterasdk.core.setup.Setup`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Global Admin Setup APIs

**static default\_settings**()

**get\_replication\_candidates**()

**get\_setup\_status**()

**init\_application\_server**(*ipaddr, secret*)

Initialize a CTERA Portal Application Server.

**Parameters**

- **ipaddr** (*str*) – The CTERA Portal master server IP address
- **secret** (*str*) – A password or a PEM-encoded private key

**init\_master**(*name, email, first\_name, last\_name, password, domain*)  
Initialize the CTERA Portal master server.

#### Parameters

- **name** (*str*) – User name for the new user
- **email** (*str*) – E-mail address of the new user
- **first\_name** (*str*) – The first name of the new user
- **last\_name** (*str*) – The last name of the new user
- **password** (*str*) – Password for the new user
- **domain** (*str*) – The domain suffix for CTERA Portal

**init\_replication\_server**(*ipaddr, secret, replicate\_from=None*)  
Initialize a CTERA Portal Database Replication Server.

#### Parameters

- **ipaddr** (*str*) – The CTERA Portal master server IP address
- **secret** (*str*) – A password or a PEM-encoded private key
- **replicate\_from** (*str*) – The name of a CTERA Portal server to replicate from

```
class cterasdk.core.setup.SetupWizardStatusMonitor(portal, retries=60, seconds=5)
    Bases: object
    wait(stage)
```

### cterasdk.core.startup module

```
class cterasdk.core.startup.Startup(portal)
    Bases: cterasdk.core.base_command.BaseCommand
    Server Startup APIs
    Started = 'Started'
    status()
        Get the server startup status
    wait(retries=120, seconds=5)
        Wait for server startup
```

### cterasdk.core.syslog module

```
class cterasdk.core.syslog.Syslog(portal)
    Bases: cterasdk.core.base_command.BaseCommand
    Portal Syslog Management APIs
    disable()
    enable(server, port=514, protocol='UDP', min_severity='info')
        Enable Syslog
```

**Parameters**

- **server** (*str*) – Syslog server address
- **port** (*int, optional*) – Syslog server port
- **protocol** (`cterasdk.core.enum.IPProtocol`, *optional*) – Syslog server IP protocol
- **min\_severity** (`cterasdk.core.enum.Severity`, *optional*) – Minimum log severity to forward

**get\_configuration()**

Retrieve the syslog server configuration

**is\_enabled()**

Check if forwarding log messages over syslog is enabled

**modify**(*server=None, port=None, protocol=None, min\_severity=None*)

Modify Syslog log forwarding configuration

**Parameters**

- **server** (*str*) – Syslog server address
- **port** (*int, optional*) – Syslog server port
- **protocol** (`cterasdk.core.enum.IPProtocol`, *optional*) – Syslog server IP protocol
- **min\_severity** (`cterasdk.core.enum.Severity`, *optional*) – Minimum log severity to forward

**cterasdk.core.taskmgr module**

**class** `cterasdk.core.taskmgr.Task`(*CTERAHost, ref, retries=10, seconds=1*)

Bases: `cterasdk.lib.task_manager_base.TaskBase`

**get\_task\_status()**

**class** `cterasdk.core.taskmgr.Tasks`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Portal Background Task APIs

**status**(*ref*)

Get background task status

**Parameters** **ref** (*str*) – Task reference

**wait**(*ref, retries=100, seconds=1*)

Wait for background task to complete

**Parameters**

- **ref** (*str*) – Task reference
- **retries** (*int, optional*) – Number of retries when sampling the task status, defaults to 100
- **seconds** (*int, optional*) – Number of seconds to wait between retries, defaults to 1



**cterasdk.core.templates module**

**class** cterasdk.core.templates.**TemplateAutoAssignPolicy**(portal)

Bases: [cterasdk.core.base\\_command.BaseCommand](#)

**apply\_changes**(wait=False)

Apply provisioning changes.

**Parameters** **wait** (*bool, optional*) – Wait for all changes to apply, defaults to *False*

**get\_policy**()

Get templates auto assignment policy

**set\_policy**(rules, apply\_default=None, default=None, apply\_changes=True)

Set templates auto assignment policy

**Parameters**

- **rules** (*list[cterasdk.common.types.PolicyRule]*) – List of policy rules
- **apply\_default** (*bool, optional*) – If no match found, apply default template. If not passed, the current config will be kept
- **default** (*str, optional*) – Name of a template to assign if no match found. Ignored unless the *apply\_default* is set to *True*
- **apply\_changes** (*bool, optional*) – Apply changes upon update, defaults to *True*

**class** cterasdk.core.templates.**Templates**(portal)

Bases: [cterasdk.core.base\\_command.BaseCommand](#)

Portal Configuration Template APIs

**add**(name, description=None, include\_sets=None, exclude\_sets=None, apps=None, backup\_schedule=None, versions=None, scripts=None, cli\_commands=None)

Add a Configuration Template

**Parameters**

- **name** (*str*) – Name of the template
- **description** (*str, optional*) – Template description
- **include\_sets** (*list[cterasdk.common.types.FilterBackupSet], optional*) – List of backup sets to include
- **exclude\_sets** (*list[cterasdk.common.types.FilterBackupSet], optional*) – List of backup sets to exclude
- **apps** (*list[cterasdk.common.enum.Application], optional*) – List of applications to back up
- **backup\_schedule** (*cterasdk.common.types.TaskSchedule, optional*) – Backup schedule
- **versions** (*list[cterasdk.core.types.PlatformVersion], optional*) – List of platforms and their associated versions. Pass *None* to inherit the default settings from the Global Administration Portal
- **scripts** (*list[cterasdk.core.types.TemplateScript], optional*) – Scripts to execute after logon, before or after backup
- **cli\_commands** (*list[str], optional*) – Template CLI commands to execute

**by\_name**(*names, include=None*)  
Get Templates by their names

**Parameters**

- **names** (*list[str], optional*) – List of names of templates
- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[cterasdk.core.query.FilterBuilder], optional*) – List of additional filters, defaults to None

**Returns** Iterator for all matching Templates

**Return type** *cterasdk.lib.iterator.Iterator*

**default** = ['name']

**delete**(*name*)  
Delete a Configuration Template

**Parameters** **name** (*str*) – Name of the template

**get**(*name, include=None*)  
Get a Configuration Template

**Parameters**

- **name** (*str*) – Name of the template
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**list\_templates**(*include=None, filters=None*)  
List Configuration Templates.

To retrieve templates, you must first browse the tenant, using: *GlobalAdmin.portals.browse()*

**Parameters**

- **include** (*list[str], optional*) – List of fields to retrieve, defaults to ['name']
- **filters** (*list[], optional*) – List of additional filters, defaults to None

**remove\_default**(*name, wait=False*)  
Set a Configuration Template not to be the default template

**Parameters**

- **name** (*str*) – Name of the template
- **wait** (*bool, optional*) – Wait for all changes to apply, defaults to *False*

**set\_default**(*name, wait=False*)  
Set a Configuration Template as the default template

**Parameters**

- **name** (*str*) – Name of the template
- **wait** (*bool, optional*) – Wait for all changes to apply, defaults to *False*

**cterasdk.core.types module**

**class** cterasdk.core.types.**ADDomainIDMapping**(*domain, start, end*)

Bases: [cterasdk.common.object.Object](#)

Base Class for Directory Service ID Mapping

**Variables** **domain** (*str*) – The domain flat name

**Parameters**

- **start** (*int*) – The minimum id to use for mapping
- **end** (*int*) – The maximum id to use for mapping

**class** cterasdk.core.types.**AccessControlEntry**(*account, role*)

Bases: tuple

Tuple holding a Portal account and its respective permission

**property** **account**

The Portal group or user account

**property** **role**

The group or user role

**class** cterasdk.core.types.**AccessControlRule**(*group, role*)

Bases: [cterasdk.common.object.Object](#)

**class** cterasdk.core.types.**AmazonS3**(*bucket, access\_key, secret\_key, endpoint='s3.amazonaws.com',  
https=True, direct=True*)

Bases: [cterasdk.core.types.HTTPBucket](#)

**to\_server\_object**()

**class** cterasdk.core.types.**AzureBlob**(*bucket, access\_key, secret\_key, endpoint='core.windows.net',  
https=True, direct=True*)

Bases: [cterasdk.core.types.HTTPBucket](#)

**to\_server\_object**()

**class** cterasdk.core.types.**Bucket**(*bucket, driver*)

Bases: object

**to\_server\_object**()

**class** cterasdk.core.types.**CloudFSFolderFindingHelper**(*name, owner*)

Bases: tuple

Tuple holding the name and owner couple to search for folders

**property** **name**

The name of the CloudFS folder

**property** **owner**

The name of the owner of the CloudFS folder

**class** cterasdk.core.types.**DomainControllers**(*primary=None, secondary=None*)

Bases: object

**property** **primary**

**property** **secondary**

**class** cterasdk.core.types.**GenericS3**(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)

Bases: [cterasdk.core.types.S3Compatible](#)

**class** `cterasdk.core.types.Google`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)  
Bases: `cterasdk.core.types.S3Compatible`

**class** `cterasdk.core.types.GroupAccount`(*name, directory=None*)

Bases: `cterasdk.core.types.PortalAccount`

**property** `account_type`

The Portal Account Type

Return `cterasdk.core.enum.PortalAccountType` The Portal Account Type

**class** `cterasdk.core.types.HTTPBucket`(*bucket, driver, access\_key, secret\_key, endpoint, https, direct=False*)

Bases: `cterasdk.core.types.Bucket`

**class** `cterasdk.core.types.ICOS`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)

Bases: `cterasdk.core.types.S3Compatible`

**class** `cterasdk.core.types.NetAppStorageGRID`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False, tags=False*)

Bases: `cterasdk.core.types.S3Compatible`

`to_server_object()`

**class** `cterasdk.core.types.Nutanix`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)

Bases: `cterasdk.core.types.S3Compatible`

**class** `cterasdk.core.types.PlanCriteriaBuilder`

Bases: `object`

`Type = 'PlanCriteria'`

`static billing_id()`

`static comment()`

`static company()`

`static first_name()`

`static last_name()`

`static role()`

`static user_groups()`

`static username()`

**class** `cterasdk.core.types.PlatformVersion`(*name, version*)

Bases: `tuple`

Tuple holding the platform name and version

**property** `name`

The name of the platform

**property** `version`

The version identifier

**class** `cterasdk.core.types.PortalAccount`(*name, directory=None*)

Bases: `abc.ABC`

Base Class for Portal Account

**Variables**

- `name` (*str*) – The user name

- **directory** (*str*) – The fully-qualified name of the user directory, defaults to None

**property account\_type**

The Portal Account Type

**Return** `cterasdk.core.enum.PortalAccountType` The Portal Account Type

**static from\_collaborator**(*collaborator*)

**property is\_local**

Is the account local

**Return bool** True if the account if local, otherwise False

**class** `cterasdk.core.types.S3Compatible`(*bucket, driver, access\_key, secret\_key, endpoint, https, direct*)

Bases: `cterasdk.core.types.HTTPBucket`

**to\_server\_object**()

**class** `cterasdk.core.types.Scality`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)

Bases: `cterasdk.core.types.S3Compatible`

**class** `cterasdk.core.types.ShareRecipient`(*account, account\_type, two\_factor=False*)

Bases: `object`

Class Representing a Collaboration Share Recipient

**static domain\_group**(*group\_account*)

Share with a domain group

**Parameters** `group_account` (`GroupAccount`) – A domain group account

**static domain\_user**(*user\_account*)

Share with a domain user

**Parameters** `user_account` (`UserAccount`) – A domain user account

**expire\_in**(*days*)

Set share to expire after (days)

**Parameters** `days` (*int*) – The number of days the share will remain accessible

**expire\_on**(*expiration\_date*)

Set the share expiration date

**Parameters** `expire_on` (*str*) – The expiration date (%Y-%m-%d)

**static external**(*email, two\_factor=False*)

Share with an external user

**Parameters**

- **email** (*str*) – The email address of the recipient
- **two\_factor** (*bool*) – Require two factor authentication over e-mail

**static local\_group**(*group\_account*)

Share with a local group

**Parameters** `group_account` (`GroupAccount`) – A local group account

**static local\_user**(*user\_account*)

Share with a local user

**Parameters** `user_account` (`UserAccount`) – A local user account

**no\_access()**

Deny access

**preview\_only()**

Grant preview only access

**read\_only()**

Grant read only access

**read\_write()**

Grant read write access

**class** cterasdk.core.types.**TemplateCriteriaBuilder**

Bases: object

**Type = 'DeviceCriteria'**

**static groups()**

**static hostname()**

**static name()**

**static os()**

**static owner()**

**static plan()**

**static type()**

**static version()**

**class** cterasdk.core.types.**TemplateScript**(*platform*)

Bases: object

**after\_backup**(*after\_backup*)

Set the post backup script

**Parameters** **after\_backup** (*str*) – A string or path to the script file

**after\_logon**(*after\_logon*)

Set the post logon script

**Parameters** **after\_logon** (*str*) – A string or path to the script file

**before\_backup**(*before\_backup*)

Set the pre backup script

**Parameters** **before\_backup** (*str*) – A string or path to the script file

**static linux()**

Configure Windows Scripts

**static mac()**

Configure Windows Scripts

**property platform**

**to\_server\_object()**

**static windows()**

Configure Windows Scripts

**class** cterasdk.core.types.**UserAccount**(*name, directory=None*)

Bases: [cterasdk.core.types.PortalAccount](#)

**property account\_type**

The Portal Account Type

**Return** `cterasdk.core.enum.PortalAccountType` The Portal Account Type

**class** `cterasdk.core.types.Wasabi`(*bucket, access\_key, secret\_key, endpoint, https=False, direct=False*)  
 Bases: `cterasdk.core.types.S3Compatible`

**cterasdk.core.users module**

**class** `cterasdk.core.users.Users`(*portal*)

Bases: `cterasdk.core.base_command.BaseCommand`

Portal User Management APIs

**add**(*name, email, first\_name, last\_name, password, role, company=None, comment=None, password\_change=False*)  
 Create a local user account

**Parameters**

- **name** (*str*) – User name for the new user
- **email** (*str*) – E-mail address of the new user
- **first\_name** (*str*) – The first name of the new user
- **last\_name** (*str*) – The last name of the new user
- **password** (*str*) – Password for the new user
- **role** (`cterasdk.core.enum.Role`) – User role of the new user
- **company** (*str, optional*) – The name of the company of the new user, defaults to None
- **comment** (*str, optional*) – Additional comment for the new user, defaults to None
- **password\_change** (*variable, optional*) – Require the user to change the password on the first login. Pass `datetime.date` for a specific date, integer for days from creation, or `True` for immediate, defaults to `False`

**apply\_changes**(*wait=False*)

Apply provisioning changes.

**Parameters** *wait* (*bool, optional*) – Wait for all changes to apply**default** = ['name']

**delete**(*user*)

Delete a user

**Parameters** *user* (`cterasdk.core.types.UserAccount`) – the user account

**get**(*user\_account, include=None*)

Get a user account

**Parameters**

- **user\_account** (`cterasdk.core.types.UserAccount`) – User account, including the user directory and user name
- **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** The user account, including the requested fields

**list\_domain\_users**(*domain, include=None*)

List all the users in the domain

**Parameters** **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** Iterator for all the domain users

**Return type** *cterasdk.lib.iterator.Iterator*

**list\_domains**()

List all domains

**Return list** List of all domains

**list\_local\_users**(*include=None*)

List all local users

**Parameters** **include** (*list[str]*) – List of fields to retrieve, defaults to ['name']

**Returns** Iterator for all local users

**Return type** *cterasdk.lib.iterator.Iterator*

**modify**(*current\_username, new\_username=None, email=None, first\_name=None, last\_name=None, password=None, role=None, company=None, comment=None*)

Modify a local user account

**Parameters**

- **current\_username** (*str*) – The current user name
- **new\_username** (*str, optional*) – New name
- **email** (*str, optional*) – E-mail address
- **first\_name** (*str, optional*) – First name
- **last\_name** (*str, optional*) – Last name
- **password** (*str, optional*) – Password
- **role** (*cterasdk.core.enum.Role, optional*) – User role
- **company** (*str, optional*) – Company name
- **comment** (*str, optional*) – Comment

### cterasdk.core.zones module

**class** *cterasdk.core.zones.Zones*(*portal*)

Bases: *cterasdk.core.base\_command.BaseCommand*

Portal Zones APIs

**add**(*name, policy\_type='selectedFolders', description=None*)

Add a new zone

**Parameters**

- **name** (*str*) – The name of the new zone
- **policy\_type** (*cterasdk.core.enum.PolicyType, optional*) – Policy type of the new zone, defaults to *cterasdk.core.enum.PolicyType.SELECT*
- **description** (*str, optional*) – The description of the new zone



**add\_devices**(*name, device\_names*)

Add devices to a zone

**Parameters**

- **name** (*str*) – The name of the zone to add devices to
- **device\_names** (*list[str]*) – The names of the devices to add to the zone

**add\_folders**(*name, folder\_finding\_helpers*)

Add the folders to the zone

**Parameters**

- **name** (*str*) – The name of the zone
- **folder\_finding\_helpers** (*list[cterasdk.core.types.CloudFSFolderFindingHelper]*) – List of folder names and owners

**delete**(*name*)

Delete a zone

**Parameters** **name** (*str*) – The name of the zone to delete

**get**(*name*)

Get zone by name

**Parameters** **name** (*str*) – The name of the zone to get

**Returns** The requested zone

### 3.1.5 cterasdk.edge package

#### 3.1.5.1 Subpackages

##### cterasdk.edge.files package

##### Submodules

##### cterasdk.edge.files.browser module

**class** cterasdk.edge.files.browser.**FileBrowser**(*Gateway*)

Bases: object

Gateway File Browser APIs

**copy**(*src, dest, overwrite=False*)

Copy a file or a folder

**Parameters**

- **src** (*str*) – Source file or folder path
- **dest** (*str*) – Destination folder path
- **overwrite** (*bool, optional*) – Overwrite on conflict, defaults to False

**delete**(*path*)

Delete a file

**Parameters** **path** (*str*) – The file's path on the gateway

**download**(*path*, *destination=None*)

Download a file

**Parameters**

- **path** (*str*) – The file path on the Edge Filer
- **destination** (*str*, *optional*) – File destination, if it is a directory, the original filename will be kept, defaults to the default directory

**download\_as\_zip**(*cloud\_directory*, *files*, *destination=None*)

Download a list of files and/or directories from a cloud folder as a ZIP file

**Warning:** The list of files is not validated. The ZIP file will include only the existing files and directories

**Parameters**

- **cloud\_directory** (*str*) – Path to the cloud directory
- **files** (*list[str]*) – List of files and/or directories in the cloud folder to download
- **destination** (*str*, *optional*) – File destination, if it is a directory, the filename will be calculated, defaults to the default directory

**static ls**(*\_path*)

**mkdir**(*path*, *recurse=False*)

Create a new directory

**Parameters**

- **path** (*str*) – The path of the new directory
- **recurse** (*bool*, *optional*) – Create subdirectories if missing, defaults to False

**static mkpath**(*path*)

**move**(*src*, *dest*, *overwrite=False*)

Move a file or a folder

**Parameters**

- **src** (*str*) – Source file or folder path
- **dest** (*str*) – Destination folder path
- **overwrite** (*bool*, *optional*) – Overwrite on conflict, defaults to False

**openfile**(*path*)

Obtain a file handle

**Parameters path** (*str*) – The file path on the Edge Filer

**upload**(*file\_path*, *server\_path*)

Upload a file

**Parameters**

- **file\_path** (*str*) – Path to the local file to upload
- **server\_path** (*str*) – Path to the directory to upload the file to

### cterasdk.edge.files.copy module

cterasdk.edge.files.copy.**copy**(*CTERAHost, src, dest, overwrite*)

### cterasdk.edge.files.file\_access module

**class** cterasdk.edge.files.file\_access.**FileAccess**(*ctera\_host*)  
Bases: *cterasdk.lib.file\_access\_base.FileAccessBase*

### cterasdk.edge.files.mkdir module

**exception** cterasdk.edge.files.mkdir.**Forbidden**(*message=None, instance=None, \*\*kwargs*)  
Bases: *cterasdk.exception.CTERAException*

**exception** cterasdk.edge.files.mkdir.**ItemExists**(*message=None, instance=None, \*\*kwargs*)  
Bases: *cterasdk.exception.CTERAException*

cterasdk.edge.files.mkdir.**mkdir**(*ctera\_host, path, recurse=False*)

### cterasdk.edge.files.move module

cterasdk.edge.files.move.**move**(*CTERAHost, src, dest, overwrite*)

### cterasdk.edge.files.path module

**class** cterasdk.edge.files.path.**CTERAPath**(*relativepath, basepath*)  
Bases: *object*

**encoded\_fullpath**()

**encoded\_parent**()

**fullpath**()

**joinpath**(*path*)

**name**()

**parent**()

**parts**()

### cterasdk.edge.files.rm module

cterasdk.edge.files.rm.**delete**(*ctera\_host, path*)

### 3.1.5.2 Submodules

#### cterasdk.edge.afp module

**class** `cterasdk.edge.afp.AFP(gateway)`  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Gateway AFP APIs

**disable()**  
Disable AFP

**is\_disabled()**  
Check if the AFP server is disabled

#### cterasdk.edge.aio module

**class** `cterasdk.edge.aio.AIO(gateway)`  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Gateway AIO APIs

**disable()**  
Disable AIO

**enable()**  
Enable AIO

**is\_enabled()**  
Is AIO enabled

**Returns** True is AIO is enabled, else False

**Return type** bool

#### cterasdk.edge.array module

**class** `cterasdk.edge.array.Array(gateway)`  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Gateway Array APIs

**add(*array\_name, level, members=None*)**  
Add a new array

**Parameters**

- **array\_name** (*str*) – Name for the new array
- **level** (`RAIDLevel`) – RAID level
- **members** (*list(str)*) – Members of the array. If not specified, the system will try to create an array using all available drives

**delete(*array\_name*)**  
Delete an array

**Parameters** **name** (*str*) – The name of the array to delete

**delete\_all()**  
Delete all arrays

**get**(*name=None*)

Get Array. If an array name was not passed as an argument, a list of all arrays will be retrieved

**Parameters** *name* (*str, optional*) – Name of the array

### cterasdk.edge.audit module

**class** `cterasdk.edge.audit.Audit`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Audit configuration APIs

**Variables** `defaultAuditEvents` (`list[cterasdk.edge.enum.AuditEvents]`) – Default audit events

`defaultAuditEvents` = ['WD', 'AD', 'WEA', 'DC', 'WA', 'DE', 'WDAC', 'WO']

**disable**()

Disable Gateway Audit log

**enable**(*path, auditEvents=None, logKeepPeriod=30, maxLogKBSize=102400, maxRotateTime=1440, includeAuditLogTag=True, humanReadableAuditLog=False*)

Enable Gateway Audit log

#### Parameters

- **path** (*str*) – Path to save the audit log
- **auditEvents** (`list[cterasdk.edge.enum.AuditEvents]`, *optional*) – List of audit event types to save, defaults to `Audit.defaultAuditEvents`
- **logKeepPeriod** (*int, optional*) – Period to key the logs in days, defaults to 30
- **maxLogKBSize** (*int, optional*) – The maximum size of the log file in KB, defaults to 102400 (100 MB)
- **maxRotateTime** (*int, optional*) – The maximal time before rotating the log file in Minutes, defaults to 1440 (24 hours)
- **includeAuditLogTag** (*bool, optional*) – Include audit log tag, defaults to True
- **humanReadableAuditLog** (*bool, optional*) – Human readable audit log, defaults to False

### cterasdk.edge.backup module

**exception** `cterasdk.edge.backup.AttachEncrypted`(*encryptionMode, encryptedFolderKey, passPhraseSalt*)

Bases: `cterasdk.exception.CTERAException`

Attach Encrypted exception

**class** `cterasdk.edge.backup.AttachRC`

Bases: `object`

`CheckCodeInCorrect` = 'CheckCodeInCorrect'

`ClocksOutOfSync` = 'ClocksOutOfSync'

`InternalServerError` = 'InternalServerError'

`IsEncrypted` = 'IsEncrypted'

```
    NotFound = 'NotFound'
    OK = 'OK'
    PermissionDenied = 'PermissionDenied'
class cterasdk.edge.backup.Backup(gateway)
    Bases: cterasdk.edge.base_command.BaseCommand
    Gateway backup configuration APIs
    configure(passphrase=None)
        Gateway backup configuration
        Parameters passphrase (str, optional) – Passphrase for the backup, defaults to None
    is_configured()
        Is Backup configured
        Return bool True if backup is configured, else False
    start()
        Start backup
    suspend()
        Suspend backup
    unsuspend()
        Unsuspend backup
class cterasdk.edge.backup.BackupFiles(gateway)
    Bases: cterasdk.edge.base_command.BaseCommand
    ALL_FILES = 'All File Types'
    unselect_all()
        Unselect all files from backup
exception cterasdk.edge.backup.ClocksOutOfSync
    Bases: cterasdk.exception.CTERAException
    Clocks Out of Sync exception
class cterasdk.edge.backup.CreateFolderRC
    Bases: object
    FolderAlreadyExists = 'FolderAlreadyExists'
    InternalServerError = 'InternalServerError'
    OK = 'OK'
    PermissionDenied = 'PermissionDenied'
class cterasdk.edge.backup.EncryptionMode
    Bases: object
    Encryption mode types
    Variables
        • Recoverable (str) – Recoverable key encryption mode
        • Secret (str) – Secret key encryption mode
    Recoverable = 'RecoverableKeyEncryption'
    Secret = 'SecretKeyEncryption'
```

**exception** `cterasdk.edge.backup.IncorrectPassphrase`

Bases: `cterasdk.exception.CTERAException`

Incorrect Passphrase exception

**exception** `cterasdk.edge.backup.NotFound`(*message=None, instance=None, \*\*kwargs*)

Bases: `cterasdk.exception.CTERAException`

Not found exception

### `cterasdk.edge.base_command` module

**class** `cterasdk.edge.base_command.BaseCommand`(*gateway*)

Bases: `object`

Base class for all Gateway API classes

**session**()

### `cterasdk.edge.cache` module

**class** `cterasdk.edge.cache.Cache`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway cache configuration

**disable**()

Disable caching

**enable**()

Enable caching

**force\_eviction**()

Force eviction

**is\_enabled**()

**pin**(*path*)

Pin a folder

**Parameters** *path* (*str*) – Directory path

**pin\_all**()

Pin all folders

**pin\_exclude**(*path*)

Exclude a sub-folder from a pinned folder

**Parameters** *path* (*str*) – Directory path

**remove\_pin**(*path*)

Remove a pin from a previously pinned folder

**Parameters** *path* (*str*) – Directory path

**unpin\_all**()

Remove all folder pins

### cterasdk.edge.cli module

**class** cterasdk.edge.cli.CLI(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway CLI APIs

**run\_command**(*cli\_command*)

Run a CLI command

**Parameters** *cli\_command* (*str*) – The CLI command to run on the gateway

**Return** *str* The response of the gateway

### cterasdk.edge.config module

**class** cterasdk.edge.config.Config(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

General gateway configuraion

**disable\_wizard**()

Disable the first time wizard

**enable\_wizard**()

Enable the first time wizard

**export**(*destination=None*)

Export the Edge Filer configuration

**Parameters** *destination* (*str, optional*) – File destination, defaults to the default directory

**get\_hostname**()

Get the hostname of the gateway

**Return** *str* The hostname of the gateway

**get\_location**()

Get the location of the gateway

**Return** *str* The location of the gateway

**import\_config**(*config, exclude=None*)

Import the Edge Filer configuration

**Parameters**

- **config** (*str*) – A string or a path to the Edge Filer configuration file
- **delete\_attrs** (*list[str], optional*) – List of configuration properties to exclude from import

**is\_wizard\_enabled**()

Get the current configuration of the first time wizard

**Return** *bool* True if the first time wizard is enabled, else False

**load\_config**(*config*)

Load the Edge Filer configuration

**Parameters** *config* (*str*) – A string or a path to the Edge Filer configuration file

**set\_hostname**(*hostname*)

Set the hostname of the gateway



**Parameters** `hostname` (*str*) – New hostname to set

**Return** *str* The new hostname

`set_location(location)`

Set the location of the gateway

**Parameters** `location` (*str*) – New location to set

**Return** *str* The new location

### `cterasdk.edge.connection` module

`cterasdk.edge.connection.test(CTERAHost)`

`cterasdk.edge.connection.test_application(CTERAHost)`

`cterasdk.edge.connection.test_network(CTERAHost)`

### `cterasdk.edge.decorator` module

`cterasdk.edge.decorator.authenticated(function)`

`cterasdk.edge.decorator.is_nosession(function, path)`

### `cterasdk.edge.directoryservice` module

**class** `cterasdk.edge.directoryservice.DirectoryService(gateway)`

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Active Directory configuration APIs

**advanced\_mapping**(*domain, start, end*)

Configure advanced mapping

**Parameters**

- **domain** (*str*) – The active directory domain
- **start** (*int*) – The minimum id to use for mapping
- **end** (*int*) – The maximum id to use for mapping

**connect**(*domain, username, password, ou=None, check\_connection=False*)

Connect the Gateway to an Active Directory

**Parameters**

- **domain** (*str*) – The active directory domain to connect to
- **username** (*str*) – The user name to use when connecting to the active directory services
- **password** (*str*) – The password to use when connecting to the active directory services
- **ou** (*str, optional*) – The OU path to use when connecting to the active directory services, defaults to None
- **check\_connection** (*bool, optional*) – Check connectivity before attempting to connect to directory services, defaults to *False*

**connected()**

Get the Active Directory join status

**disconnect()**

Disconnect from Active Directory Service

**domains()**

Get all domains

**Return list(str)** List of names of all discovered domains

**get\_connected\_domain()**

Get the connected domain information

**Return cterasdk.common.object.Object**

**get\_static\_domain\_controller()**

Retrieve the static domain controller configuration

**Returns** A FQDN, hostname or ip address of the domain controller

**Return type** str

**remove\_static\_domain\_controller()**

Delete the static domain controller configuration

**set\_static\_domain\_controller(dc)**

Configure the Gateway to use a static domain controller

**Parameters dc (str)** – The FQDN, hostname or ip address of the domain controller

**Returns** The FQDN, hostname or ip address of the domain controller

**Return type** str

**cterasdk.edge.drive module**

**class cterasdk.edge.drive.Drive(gateway)**

Bases: [cterasdk.edge.base\\_command.BaseCommand](#)

Gateway Drive APIs

**format(name)**

Format a drive

**Parameters name (str)** – The name of the drive to format

**format\_all()**

Format all drives

**get(name=None)**

Get Drive. If a drive name was not passed as an argument, a list of all drives will be retrieved

**Parameters name (str, optional)** – Name of the drive

**get\_status(name=None)**

Get drive status. If a drive name was not passed as an argument, a list of all drives will be retrieved

**Parameters name (str)** – Name of the drive

**cterasdk.edge.enum module****class** cterasdk.edge.enum.Acl

Bases: object

ACL types

**Variables**

- **WindowsNT** (*str*) – Windows NT ACL Mode
- **OnlyAuthenticatedUsers** (*str*) – Authenticated Users ACL Mode

**OnlyAuthenticatedUsers** = 'authenticated'**WindowsNT** = 'winAclMode'**class** cterasdk.edge.enum.AuditEvents

Bases: object

Audit log event types

**Variables**

- **ListFolderReadData** (*str*) – List Folder Read Data
- **CreateFilesWriteData** (*str*) – Create Files Write Data
- **CreateFoldersAppendData** (*str*) – Create Folders Append Data
- **ReadExtendedAttributes** (*str*) – Read Extended Attributes
- **WriteExtendedAttributes** (*str*) – Write Extended Attributes
- **TraverseFolderExecuteFile** (*str*) – Traverse Folder Execute File
- **DeleteSubfoldersAndFiles** (*str*) – Delete Subfolders And Files
- **WriteAttributes** (*str*) – Write Attributes
- **Delete** (*str*) – Delete
- **ChangePermissions** (*str*) – Change Permissions
- **ChangeOwner** (*str*) – Change Owner

**ChangeOwner** = 'WO'**ChangePermissions** = 'WDAC'**CreateFilesWriteData** = 'WD'**CreateFoldersAppendData** = 'AD'**Delete** = 'DE'**DeleteSubfoldersAndFiles** = 'DC'**ListFolderReadData** = 'RD'**ReadExtendedAttributes** = 'REA'**TraverseFolderExecuteFile** = 'X'**WriteAttributes** = 'WA'**WriteExtendedAttributes** = 'WEA'

**class** cterasdk.edge.enum.BackupConfStatusID

Bases: object

Status of backup configuration

**Variables**

- **NotInitialized** (*str*) – Backup configuration was not initialized
- **Configuring** (*str*) – Backup is being configured
- **Attaching** (*str*) – Backup configuration is Attaching
- **Attached** (*str*) – Backup configuration is Attached
- **NoFolder** (*str*) – No Folder for backup
- **WrongPassword** (*str*) – Wrong password used
- **Failed** (*str*) – Backup configuration failed
- **Unsubscribed** (*str*) – Unsubscribed to backup
- **Unlicensed** (*str*) – Unlicensed” to backup
- **ClocksOutOfSync** (*str*) – Clocks are out of sync
- **GetFoldersList** (*str*) – Get folders list

**Attached** = 'Attached'

**Attaching** = 'Attaching'

**ClocksOutOfSync** = 'ClocksOutOfSync'

**Configuring** = 'Configuring'

**Failed** = 'Failed'

**GetFoldersList** = 'GetFoldersList'

**NoFolder** = 'NoFolder'

**NotInitialized** = 'NotInitialized'

**Unlicensed** = 'Unlicensed'

**Unsubscribed** = 'Unsubscribed'

**WrongPassword** = 'WrongPassword'

**class** cterasdk.edge.enum.CIFS Packet Signing

Bases: object

CIFS Packet signing options

**Variables**

- **Disabled** (*str*) – CIFS Packet signing is disabled
- **IfClientAgrees** (*str*) – Use CIFS Packet signing is client agrees
- **Required** (*str*) – Require CIFS Packet signing

**Disabled** = 'Disabled'

**IfClientAgrees** = 'If client agrees'

**Required** = 'Required'

```
class cterasdk.edge.enum.ClientSideCaching
```

```
    Bases: object
```

```
    Client side caching types
```

```
        Variables
```

- **Manual** (*str*) – Manual client side caching
- **Documents** (*str*) – Documents client side caching
- **Disabled** (*str*) – Client side caching disabled

```
    Disabled = 'disable'
```

```
    Documents = 'documents'
```

```
    Manual = 'manual'
```

```
class cterasdk.edge.enum.FileAccessMode
```

```
    Bases: object
```

```
    File Access Mode
```

```
        Variables
```

- **RW** (*str*) – Read Write
- **RO** (*str*) – Read Only
- **NA** (*str*) – None

```
    NA = 'None'
```

```
    RO = 'ReadOnly'
```

```
    RW = 'ReadWrite'
```

```
class cterasdk.edge.enum.IPProtocol
```

```
    Bases: object
```

```
    IP Protocol
```

```
        Variables
```

- **TCP** (*str*) – TCP Protocol
- **UDP** (*str*) – UDP Protocol

```
    TCP = 'TCP'
```

```
    UDP = 'UDP'
```

```
class cterasdk.edge.enum.License
```

```
    Bases: object
```

```
    Gateway license types
```

```
        Variables
```

- **EV4** (*str*) – EV4 license
- **EV8** (*str*) – EV8 license
- **EV16** (*str*) – EV16 license
- **EV32** (*str*) – EV32 license
- **EV64** (*str*) – EV64 license

- **EV128** (*str*) – EV128 license

**EV128** = 'EV128'

**EV16** = 'EV16'

**EV32** = 'EV32'

**EV4** = 'EV4'

**EV64** = 'EV64'

**EV8** = 'EV8'

**class** cterasdk.edge.enum.**LocalGroup**

Bases: object

Local Group types

**Variables**

- **Administrators** (*str*) – Administrators
- **ReadOnlyAdministrators** (*str*) – Read Only Administrators
- **Everyone** (*str*) – Everyone

**Administrators** = 'Administrators'

**Everyone** = 'Everyone'

**ReadOnlyAdministrators** = 'Read Only Administrators'

**class** cterasdk.edge.enum.**Mode**

Bases: object

Enum for operational mode

**Variables**

- **Enabled** (*str*) – Operational mode enabled
- **Disabled** (*str*) – Operational mode disabled

**Disabled** = 'disabled'

**Enabled** = 'enabled'

**class** cterasdk.edge.enum.**OperationMode**

Bases: object

Gateway operation mode

**Variables**

- **Disabled** (*str*) – Gateway is Disabled
- **CachingGateway** (*str*) – Gateway is in Caching mode

**CachingGateway** = 'CachingGateway'

**Disabled** = 'Disabled'

**class** cterasdk.edge.enum.**PrincipalType**

Bases: object

ACL Principal Type

**Variables**

- **LU** (*str*) – Local User
- **LG** (*str*) – Local Group
- **DU** (*str*) – Domain User
- **DG** (*str*) – Domain Group

**DG** = 'DomainGroup'

**DU** = 'DomainUser'

**LG** = 'LocalGroup'

**LU** = 'LocalUser'

**class** cterasdk.edge.enum.**RAIDLevel**

Bases: object

RAID Levels

#### Variables

- **JBOD** (*str*) – Linear concatenation
- **RAID\_0** (*str*) – Stripe set
- **RAID\_1** (*str*) – Mirror
- **RAID\_5** (*str*) – Distributed parity
- **RAID\_6** (*str*) – Dual parity

**JBOD** = 'linear'

**LVM** = 'LVM'

**RAID\_0** = '0'

**RAID\_1** = '1'

**RAID\_5** = '5'

**RAID\_6** = '6'

**class** cterasdk.edge.enum.**SMBProtocol**

Bases: object

SMB Protocol

#### Variables

- **SMB1** (*str*) – SMB v1
- **NT1** (*str*) – SMB v1
- **SMB2\_02** (*str*) – Vista, Server 2008
- **SMB2\_10** (*str*) – Windows 7, Server 2008 R2
- **SMB3\_00** (*str*) – Windows 8, Server 2012
- **SMB3\_02** (*str*) – Windows 8.1, Server 2012 R2
- **SMB3\_11** (*str*) – Windows 10, Server 2016
- **SMB2** (*str*) – Windows 7, Server 2008
- **SMB3** (*str*) – SMB 3.1.1

**NT1** = 'NT1'

```
SMB1 = 'NT1'  
SMB2 = 'SMB2'  
SMB2_02 = 'SMB2_02'  
SMB2_10 = 'SMB2_10'  
SMB3 = 'SMB3'  
SMB3_00 = 'SMB3_00'  
SMB3_02 = 'SMB3_02'  
SMB3_11 = 'SMB3_11'
```

```
class cterasdk.edge.enum.ServicesConnectionState
```

```
Bases: object
```

```
Gateway connection status
```

#### Variables

- **ResolvingServers** (*str*) – The Edge Filer is resolving CTERA Portal servers
- **Connecting** (*str*) – The Edge Filer is in connecting to CTERA Portal
- **Attaching** (*str*) – The Edge Filer is attaching to CTERA Portal
- **Authenticating** (*str*) – The Edge Filer is authenticating to CTERA Portal
- **Disconnected** (*str*) – The Edge Filer is disconnected from CTERA Portal
- **Connected** (*str*) – The Edge Filer is connected to CTERA Portal

```
Attaching = 'Attaching'
```

```
Authenticating = 'Authenticating'
```

```
Connected = 'Connected'
```

```
Connecting = 'Connecting'
```

```
Disconnected = 'Disconnected'
```

```
ResolvingServers = 'ResolvingServers'
```

```
class cterasdk.edge.enum.Severity
```

```
Bases: object
```

```
Log severity levels
```

#### Variables

- **EMERGENCY** (*str*) – Emergency log level
- **ALERT** (*str*) – Alert log level
- **CRITICAL** (*str*) – Critical log level
- **ERROR** (*str*) – Error log level
- **WARNING** (*str*) – Warning log level
- **NOTICE** (*str*) – Notice log level
- **INFO** (*str*) – Info log level
- **DEBUG** (*str*) – Debug log level

```
ALERT = 'alert'
```



```

CRITICAL = 'critical'
DEBUG = 'debug'
EMERGENCY = 'emergency'
ERROR = 'error'
INFO = 'info'
NOTICE = 'notice'
WARNING = 'warning'

```

```
class cterasdk.edge.enum.SyncStatus
```

```
Bases: object
```

```
Gateway sync status
```

#### Variables

- **Off** (*str*) – Off
- **NotInitialized** (*str*) – Not Initialized
- **InitializingConnection** (*str*) – Initializing Connection
- **ConnectingFolders** (*str*) – Connecting Folders
- **Connected** (*str*) – Connected
- **ClocksOutOfSync** (*str*) – Clocks is Out Of Sync
- **ConnectionFailed** (*str*) – Connection Failed
- **InternalError** (*str*) – Internal Error
- **InvalidConfiguration** (*str*) – Invalid Configuration
- **VolumeUnavailable** (*str*) – Volume Unavailable
- **NoFolder** (*str*) – No Folder
- **DisconnectedPortal** (*str*) – Disconnected from Portal
- **ServiceUnavailable** (*str*) – Service is Unavailable
- **Unlicensed** (*str*) – Unlicensed
- **Synced** (*str*) – Synced
- **Syncing** (*str*) – Syncing
- **Scanning** (*str*) – Scanning
- **UpgradingDataBase** (*str*) – Upgrading Database
- **OutOfQuota** (*str*) – Out of Quota
- **RejectedByPolicy** (*str*) – Rejected by Policy
- **FailedFilesInReadOnlyFolder** (*str*) – Failed Files in Read Only Folder
- **ShouldSupportWinNtAcl** (*str*) – Should Support WinNt Acl
- **TakingSnapshot** (*str*) – Taking Snapshot
- **CatalogReadOnlyMode** (*str*) – Catalog ReadOnly Mode
- **InvalidAverageBlockSize** (*str*) – Invalid Average Block Size

```
CatalogReadOnlyMode = 'CatalogReadOnlyMode'
ClocksOutOfSync = 'ClocksOutOfSync'
Connected = 'Connected'
ConnectingFolders = 'ConnectingFolders'
ConnectionFailed = 'ConnectionFailed'
DisconnectedPortal = 'DisconnectedPortal'
FailedFilesInReadOnlyFolder = 'FailedFilesInReadOnlyFolder'
InitializingConnection = 'InitializingConnection'
InternalError = 'InternalError'
InvalidAverageBlockSize = 'InvalidAverageBlockSize'
InvalidConfiguration = 'InvalidConfiguration'
NoFolder = 'NoFolder'
NotInitialized = 'NotInitialized'
Off = 'Off'
OutOfQuota = 'OutOfQuota'
RejectedByPolicy = 'RejectedByPolicy'
Scanning = 'Scanning'
ServiceUnavailable = 'ServiceUnavailable'
ShouldSupportWinNtAcl = 'ShouldSupportWinNtAcl'
Synced = 'Synced'
Syncing = 'Syncing'
TakingSnapshot = 'TakingSnapshot'
Unlicensed = 'Unlicensed'
UpgradingDataBase = 'UpgradingDataBase'
VolumeUnavailable = 'VolumeUnavailable'

class cterasdk.edge.enum.TCPConnectRC
    Bases: object
    Open = 'Open'

class cterasdk.edge.enum.TaskStatus
    Bases: object
    Gateway task status

    Variables
        • Failed (str) – The task has failed
        • Running (str) – The task is running
        • Completed (str) – The task has completed

    Completed = 'completed'
    Failed = 'failed'
```

**Running** = 'running'

**class** cterasdk.edge.enum.Traffic

Bases: object

Traffic type

**Variables**

- **Upload** (*str*) – Upload
- **Download** (*str*) – Download

**Download** = 'Download'

**Upload** = 'Upload'

**class** cterasdk.edge.enum.VolumeStatus

Bases: object

Gateway volume status

**Variables**

- **Ok** (*str*) – Volume is ok
- **ContainsErrors** (*str*) – Volume contains errors
- **ReadOnly** (*str*) – Volume is read only
- **Corrupted** (*str*) – Volume is corrupted
- **Unknown** (*str*) – Volume status is unknown
- **Recovering** (*str*) – Volume is recovering
- **Mounting** (*str*) – Volume is mounting
- **Unmounted** (*str*) – Volume is unmounted
- **Formatting** (*str*) – Volume is formatting
- **Converting** (*str*) – Volume is converting
- **Resizing** (*str*) – Volume is resizing
- **Repairing** (*str*) – Volume is repairing
- **Checking** (*str*) – Volume is checking
- **KeyRequired** (*str*) – Volume required key
- **CheckingQuota** (*str*) – Checking volume quota

**Checking** = 'checking'

**CheckingQuota** = 'checkingQuota'

**ContainsErrors** = 'containsErrors'

**Converting** = 'converting'

**Corrupted** = 'corrupted'

**Formatting** = 'formatting'

**KeyRequired** = 'keyRequired'

**Mounting** = 'mounting'

**Ok** = 'ok'

```
ReadOnly = 'readOnly'  
Recovering = 'recovering'  
Repairing = 'repairing'  
Resizing = 'resizing'  
Unknown = 'unknown'  
Unmounted = 'unmounted'
```

### cterasdk.edge.ftp module

```
class cterasdk.edge.ftp.FTP(gateway)
```

```
    Bases: cterasdk.edge.base_command.BaseCommand
```

```
    Gateway FTP configuration APIs
```

```
    disable()  
        Disable FTP
```

```
    enable()  
        Enable FTP
```

```
    get_configuration()  
        Get the current FTP configuration
```

```
        Return cterasdk.common.object.Object
```

```
    is_disabled()  
        Check if the FTP server is disabled
```

```
    modify(allow_anonymous_ftp=None, anonymous_download_limit=None, anonymous_ftp_folder=None,  
          banner_message=None, max_connections_per_ip=None, require_ssl=None)
```

```
        Modify the FTP Configuration. Parameters that are not passed will not be affected
```

#### Parameters

- **allow\_anonymous\_ftp** (*bool, optional*) – Enable/Disable anonymous FTP downloads
- **anonymous\_download\_limit** (*int, optional*) – Limit download bandwidth of anonymous connection in KB/sec per connection. 0 for unlimited
- **anonymous\_ftp\_folder** (*str, optional*) – Anonymous FTP Directory
- **banner\_message** (*str, optional*) – FTP Banner Message
- **max\_connections\_per\_ip** (*int, optional*) – Maximum Connections per Client
- **require\_ssl** (*bool, optional*) – If True, allow only SSL/TLS connections

**cterasdk.edge.firmware module**

**class** `cterasdk.edge.firmware.Firmware`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Firmware upgrade API

**upgrade**(*file\_path*, *reboot=True*, *wait\_for\_reboot=True*)

Upgrade the Filer firmware with the provided file

**Parameters**

- **file\_path** (*str*) – Path to the local file to upload
- **reboot** (*bool*, *optional*) – Perform reboot after uploading the new firmware, defaults to True
- **wait\_for\_reboot** (*bool*, *optional*) – Wait for reboot to complete (if reboot is performed), defaults to True

**class** `cterasdk.edge.firmware.UploadTaskStatus`

Bases: `object`

**COMPLETE** = 1

**FAIL** = -1

**IN\_PROGRESS** = 0

**cterasdk.edge.groups module**

**class** `cterasdk.edge.groups.Groups`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

**add\_members**(*group*, *members*)

Add members to a group

**Parameters**

- **group** (*str*) – Name of the group
- **members** (*list*[`cterasdk.edge.types.UserGroupEntry`]) – List of users and groups to add to the group

**get**(*name=None*)

Get Group. If a group name was not passed as an argument, a list of all local groups will be retrieved

**Parameters** **name** (*str*, *optional*) – Name of the group

**remove\_members**(*group*, *members*)

Remove members from a group

**Parameters**

- **group** (*str*) – Name of the group
- **members** (*list*[`cterasdk.edge.types.UserGroupEntry`]) – List of users and groups to remove from the group

### cterasdk.edge.licenses module

**class** cterasdk.edge.licenses.Licenses(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Edge Filer License Configuration APIs

**apply**(*ctera\_license*)

Apply a license

**Parameters** *ctera\_license* (*cterasdk.edge.enum.License*) – License type

**get**()

Get the current Gateway License

**static infer**(*ctera\_license*)

**class** cterasdk.edge.licenses.LocalLicenses(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Edge Filer Local License Configuration APIs

**add**(*code*)

Install a local license. Use this option when running the Edge Filer as a standalone appliance.

**Parameters** *code* (*str*) – License code

**clear**()

Remove all local licenses

**get**()

Retrieve a list of local licenses installed

### cterasdk.edge.login module

**class** cterasdk.edge.login.Login(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

**info**()

Get login info

**login**(*username*, *password*)

**logout**()

### cterasdk.edge.logs module

**class** cterasdk.edge.logs.Logs(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Logs APIs

**Variables** *default\_include* (*list[str]*) – Default log fields - ‘severity’, ‘time’, ‘msg’, ‘more’

**default\_include** = ['severity', 'time', 'msg', 'more']

**logs**(*topic*, *include=None*, *minSeverity='info'*)

Fetch Gateway logs

**Parameters**

- **topic** (*str*) – Log Topic to fetch

- **include** (*list[str], optional*) – List of fields to include in the response, defaults to `Logs.default_include`
- **minSeverity** (`cterasdk.edge.enum.Severity`, *optional*) – Minimal log severity to fetch, defaults to `cterasdk.edge.enum.Severity.INFO`

**Returns** Log lines

**Return type** `cterasdk.lib.iterator.Iterator`

**settings**(*retention, min\_severity=None*)  
Configure log settings

**Parameters**

- **retention** (*int*) – Log retention period in days
- **min\_severity** (`cterasdk.edge.enum.Severity`, *optional*) – Minimal log severity

### cterasdk.edge.mail module

**class** `cterasdk.edge.mail.Mail`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Mail Server configuration APIs

**disable**()

Disable e-mail delivery using a custom SMTP server

**enable**(*smtp\_server, port=25, username=None, password=None, use\_tls=True*)

Enable e-mail delivery using a custom SMTP server

**Parameters**

- **smtp\_server** (*str*) – Address of the SMTP Server
- **port** (*int, optional*) – The listening port of the SMTP Server, defaults to 25
- **username** (*str, optional*) – The user name of the SMTP Server, defaults to None
- **password** (*str, optional*) – The password of the SMTP Server, defaults to None
- **use\_tls** (*bool, optional*) – Use TLS when connecting to the SMTP Server, defaults to True

### cterasdk.edge.network module

**class** `cterasdk.edge.network.Network`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Network configuration APIs

**diagnose**(*services*)

Test a TCP connection to a host over a designated port

**Parameters** **services** (*list[cterasdk.edge.types.TCPService]*) – List of services, identified by a host and a port

**Returns** A list of named-tuples including the host, port and a boolean value indicating whether TCP connection can be established

**Return type** `list[cterasdk.edge.types.TCPConnectResult]`

**enable\_dhcp()**

Enable DHCP

**get\_status()**

Retrieve the network interface status

**ifconfig()**

Retrieve the ip configuration

**ipconfig()**

Retrieve the ip configuration

**iperf**(*address, port=5201, threads=1, protocol='TCP', direction='Upload', retries=120, seconds=1*)

Invoke a network throughput test

**Parameters**

- **address** (*str*) – The host running the iperf server
- **port** (*int, optional*) – The iperf server port, defaults to 5201
- **threads** (*int, optional*) – The number of threads, defaults to 1
- **protocol** (`cterasdk.edge.enum.IPProtocol`, *optional*) – IP protocol, defaults to 'TCP'
- **direction** (`cterasdk.edge.enum.Traffic`, *optional*) – Traffic direction, defaults to 'Upload'
- **retries** (*int, optional*) – Number of retries when sampling the iperf task status, defaults to 120
- **seconds** (*int, optional*) – Number of seconds to wait between retries, defaults to 1

**Returns** A string containing the iperf output**Return type** str**reset\_mtu()**

Set the default maximum transmission unit (MTU) settings

**set\_mtu**(*mtu*)

Set a custom network maximum transmission unit (MTU)

**Parameters** **mtu** (*int*) – Maximum transmission unit**set\_static\_ipaddr**(*address, subnet, gateway, primary\_dns\_server, secondary\_dns\_server=None*)

Set a Static IP Address

**Parameters**

- **address** (*str*) – The static address
- **subnet** (*str*) – The subnet for the static address
- **gateway** (*str*) – The default gateway
- **primary\_dns\_server** (*str*) – The primary DNS server
- **secondary\_dns\_server** (*str, optional*) – The secondary DNS server, defaults to None

**set\_static\_nameserver**(*primary\_dns\_server, secondary\_dns\_server=None*)

Set the DNS Server addresses statically

**Parameters**

- **primary\_dns\_server** (*str*) – The primary DNS server



- **secondary\_dns\_server** (*str, optional*) – The secondary DNS server, defaults to None

**tcp\_connect**(*service*)

Test a TCP connection between the Gateway and the provided host address

**Parameters** **service** (*cterasdk.edge.types.TCPService*) – A service, identified by a host and a port

**Returns** A named-tuple including the host, port and a boolean value indicating whether TCP connection can be established

**Return type** *cterasdk.edge.types.TCPConnectResult*

### cterasdk.edge.nfs module

**class** *cterasdk.edge.nfs.NFS*(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway NFS configuration

**disable**()

Disable NFS

**enable**()

Enable NFS

**get\_configuration**()

Get the current NFS configuration

**Return** *cterasdk.common.object.Object*

**is\_disabled**()

Check if the NFS server is disabled

**modify**(*async\_write=None, aggregate\_writes=None, mountd\_port=None, statd\_port=None, nfsv4\_enabled=None, krb5\_enabled=None*)

Modify the FTP Configuration. Parameters that are not passed will not be affected

**Parameters**

- **async\_write** (*bool, optional*) – If True, use asynchronous writes
- **aggregate\_writes** (*bool, optional*) – If True, aggregate write requests

### cterasdk.edge.ntp module

**class** *cterasdk.edge.ntp.NTP*(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway NTP configuration

**disable**()

Disable NTP

**enable**(*servers=None*)

Enable NTP

**Parameters** **servers** (*list[str]*) – List of NTP servers address

**get\_configuration**()

**property** **servers**

### cterasdk.edge.ssh module

**class** cterasdk.edge.ssh.SSH(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Edge Filer SSH daemon APIs

**disable()**

**enable**(*public\_key=None, public\_key\_file=None, exponent=65537, key\_size=2048*)

Enable the Edge Filer's SSH daemon

#### Parameters

- **public\_key** (*str, optional*) – A PEM-encoded public key in OpenSSH format. If neither a public key nor public key file were specified, an RSA key pair will be generated automatically. The PEM-encoded private key will be saved to the default Downloads directory
- **public\_key\_file** (*str, optional*) – A path to the public key file
- **exponent** (*int, optional*) – The public exponent of the new key, defaults to 65537
- **key\_size** (*int, optional*) – The length of the modulus in bits, defaults to 2048

### cterasdk.edge.ssl module

**class** cterasdk.edge.ssl.SSL(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Edge Filer SSL APIs

**disable\_http()**

Disable HTTP access

**enable\_http()**

Enable HTTP access

**get\_storage\_ca()**

Get object storage trusted CA certificate

**import\_certificate**(*private\_key, \*certificates*)

Import the Edge Filer's web server's SSL certificate

#### Parameters

- **private\_key** (*str*) – The PEM-encoded private key, or a path to the PEM-encoded private key file
- **certificates** (*list[str]*) – The PEM-encoded certificates, or a list of paths to the PEM-encoded certificates

**import\_storage\_ca**(*certificate*)

Import the object storage trusted CA certificate

**Parameters certificate** (*str*) – The PEM-encoded certificate or a path to the PEM-encoded server certificate file

**is\_http\_disabled()**

Check if HTTP access is disabled

**is\_http\_enabled()**

Check if HTTP access is enabled

**remove\_storage\_ca()**  
Remove object storage trusted CA certificate

### cterasdk.edge.power module

**class** cterasdk.edge.power.**Boot**(*gateway, retries=60, seconds=5*)  
Bases: object

**wait()**

**class** cterasdk.edge.power.**Power**(*gateway*)  
Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Power APIs

**reboot**(*wait=False*)  
Reboot the Gateway

**Parameters** **wait** (*bool, optional*) – Wait got the reboot to complete, defaults to False

**reset**(*wait=False*)  
Reset the Gateway setting

**Parameters** **wait** (*bool, optional*) – Wait got the reset to complete, defaults to False

**shutdown()**  
Shutdown the Gateway

### cterasdk.edge.query module

**class** cterasdk.edge.query.**QueryParam**  
Bases: *cterasdk.common.object.Object*

**include\_classname()**

**increment()**

**class** cterasdk.edge.query.**QueryParamBuilder**  
Bases: object

**build()**

**countLimit**(*countLimit*)

**include**(*include*)

**put**(*key, value*)

**startFrom**(*startFrom*)

cterasdk.edge.query.**query**(*CTERAHost, path, key, value*)

cterasdk.edge.query.**show**(*CTERAHost, path, key, value*)

### cterasdk.edge.remote module

`cterasdk.edge.remote.login(Gateway, ticket)`

`cterasdk.edge.remote.obtain_ticket(Portal, device_name)`

`cterasdk.edge.remote.remote_access(Gateway, Portal)`

### cterasdk.edge.rsync module

**class** `cterasdk.edge.rsync.RSync(gateway)`

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway RSync configuration

**disable()**

Disable FTP

**enable()**

Enable FTP

**get\_configuration()**

Get the current RSync configuration

**Return** `cterasdk.common.object.Object`

**is\_disabled()**

Check if the Rsync server is disabled

**modify**(*port=None, max\_connections=None*)

Modify the RSync Configuration. Parameters that are not passed will not be affected

**Parameters**

- **port** (*int, optional*) – RSync Port
- **max\_connections** (*int, optional*) – Maximum Connections

### cterasdk.edge.services module

**class** `cterasdk.edge.services.Services(gateway)`

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Cloud Services configuration APIs

**activate**(*server, user, code, ctera\_license='EV16'*)

Activate the gateway using an activation code

**Parameters**

- **server** (*str*) – Address of the Portal
- **user** (*str*) – User for the Portal connection
- **code** (*str*) – Activation code for the Portal connection
- **ctera\_license** (`cterasdk.edge.enum.License, optional`) – CTERA License, defaults to `cterasdk.edge.enum.License.EV16`

**connect**(*server, user, password, ctera\_license='EV16'*)

Connect to a Portal.

The `connect` method will first validate the *license* argument, ensure the Gateway can establish a TCP connection over port 995 to *server* using `Gateway.tcp_connect()` and verify the Portal does not require device activation via code

#### Parameters

- **server** (*str*) – Address of the Portal
- **user** (*str*) – User for the Portal connection
- **password** (*str*) – Password for the Portal connection
- **ctera\_license** (`cterasdk.edge.enum.License`, *optional*) – CTERA License, defaults to `cterasdk.edge.enum.License.EV16`

#### `connected()`

Check if the Edge Filer is connected to CTERA Portal

#### `disable_sso()`

Disable SSO connection

#### `disconnect()`

Disconnect from the Portal

#### `enable_sso()`

Enable SSO connection

#### `get_status()`

Retrieve the cloud services connection status

#### `reconnect()`

Reconnect to the Portal

#### `sso_enabled()`

Is SSO connection enabled

**Return bool** True if SSO connection is enabled, else False

### `cterasdk.edge.session` module

```
class cterasdk.edge.session.Session(host)
```

Bases: `cterasdk.lib.session_base.SessionBase`

```
    disable_remote_access()
```

```
    enable_remote_access()
```

```
    local()
```

```
    remote()
```

```
    remote_access()
```

```
    remote_from()
```

```
    start_remote_session(remote_session)
```

```
class cterasdk.edge.session.SessionConnection(session_type, remote_from=None)
```

Bases: `cterasdk.common.object.Object`

```
class cterasdk.edge.session.SessionType
```

Bases: `object`

```
Local = 'local'
```

```
Remote = 'remote'
```

### cterasdk.edge.shares module

```
class cterasdk.edge.shares.Shares(gateway)
```

Bases: `cterasdk.edge.base_command.BaseCommand`

```
add(name, directory, acl=None, access='winAclMode', csc='manual', dir_permissions=777, comment=None,
     export_to_afp=False, export_to_ftp=False, export_to_nfs=False, export_to_pc_agent=False,
     export_to_rsync=False, indexed=False, trusted_nfs_clients=None, uid=None)
```

Add a network share.

#### Parameters

- **name** (*str*) – The share name
- **directory** (*str*) – Full directory path
- **acl** (*list*[`cterasdk.edge.types.ShareAccessControlEntry`]) – List of access control entries
- **access** (`cterasdk.edge.enum.Acl`) – The Windows File Sharing authentication mode, defaults to `winAclMode`
- **csc** (`cterasdk.edge.enum.ClientSideCaching`) – The client side caching (offline files) configuration, defaults to `manual`
- **dir\_permissions** (*int*) – Directory Permission, defaults to `777`
- **comment** (*str*) – Comment
- **export\_to\_afp** (*bool*) – Whether to enable AFP access, defaults to `False`
- **export\_to\_ftp** (*bool*) – Whether to enable FTP access, defaults to `False`
- **export\_to\_nfs** (*bool*) – Whether to enable NFS access, defaults to `False`
- **export\_to\_pc\_agent** (*bool*) – Whether to allow as a destination share for CTERA Backup Agents, defaults to `False`
- **export\_to\_rsync** (*bool*) – Whether to enable access over rsync, defaults to `False`
- **indexed** (*bool*) – Whether to enable indexing for search, defaults to `False`
- **trusted\_nfs\_clients** (*list*[`cterasdk.edge.types.NFSv3AccessControlEntry`]) – Trusted NFS v3 clients, defaults to `None`

```
add_acl(name, acl)
```

Add one or more access control entries to an existing share.

#### Parameters

- **name** (*str*) – The share name
- **acl** (*list*[`cterasdk.edge.types.ShareAccessControlEntry`]) – List of access control entries to add

```
add_screened_file_types(name, extensions)
```

Add extensions to the share's current list of blocked file extensions

#### Parameters

- **name** (*str*) – The share name

- **extensions** (*list[str]*) – List of file extensions to add

**add\_trusted\_nfs\_clients**(*name, trusted\_nfs\_clients*)

Add one or more trusted NFS client entries to an existing share.

**Parameters**

- **name** (*str*) – The share name
- **trusted\_nfs\_clients** (*list[cterasdk.edge.types.NFSv3AccessControlEntry]*) – Trusted NFS v3 clients

**block\_files**(*name, extensions*)

Configure a share to block one or more file extensions

**Parameters**

- **name** (*str*) – The share name
- **extensions** (*list[str]*) – List of file extensions to block

**delete**(*name*)

Delete a share.

**Parameters** **name** (*str*) – The share name

**get**(*name=None*)

Get Share. If a share name was not passed as an argument, a list of all shares will be retrieved :param str,optional name: Name of the share

**get\_access\_type**(*name*)

Get the network share Windows File Sharing authentication mode

**Parameters** **name** (*str*) – The share name

**get\_acl**(*name*)

Get the current access control entries from an existing share.

**Parameters** **name** (*str*) – The share name

**get\_screened\_file\_types**(*name*)

Get the share's current list of blocked file extensions

**Parameters** **name** (*str*) – The share name

**get\_trusted\_nfs\_clients**(*name*)

Get the current trusted NFS client entries from an existing share.

**Parameters** **name** (*str*) – The share name

**modify**(*name, directory=None, acl=None, access=None, csc=None, dir\_permissions=None, comment=None, export\_to\_afp=None, export\_to\_ftp=None, export\_to\_nfs=None, export\_to\_pc\_agent=None, export\_to\_rsync=None, indexed=None, trusted\_nfs\_clients=None*)

Modify an existing network share. All parameters but name are optional and default to None

**Parameters**

- **name** (*str*) – The share name
- **directory** (*str, optional*) – Full directory path
- **acl** (*list[cterasdk.edge.types.ShareAccessControlEntry], optional*) – List of access control entries
- **access** (*cterasdk.edge.enum.Acl, optional*) – The Windows File Sharing authentication mode

- **csc** (`cterasdk.edge.enum.ClientSideCaching`, *optional*) – The client side caching (offline files) configuration
- **dir\_permissions** (`int`, *optional*) – Directory Permission
- **comment** (`str`, *optional*) – Comment
- **export\_to\_afp** (`bool`, *optional*) – Whether to enable AFP access
- **export\_to\_ftp** (`bool`, *optional*) – Whether to enable FTP access
- **export\_to\_nfs** (`bool`, *optional*) – Whether to enable NFS access
- **export\_to\_pc\_agent** (`bool`, *optional*) – Whether to allow as a destination share for CTERA Backup Agents
- **export\_to\_rsync** (`bool`, *optional*) – Whether to enable access over rsync
- **indexed** (`bool`, *optional*) – Whether to enable indexing for search
- **trusted\_nfs\_clients** (`list[cterasdk.edge.types.NFSv3AccessControlEntry]`) – Trusted NFS v3 clients, defaults to None

**remove\_acl**(*name, acl*)

Remove one or more access control entries from an existing share.

**Parameters**

- **name** (`str`) – The share name
- **acl** (`list[cterasdk.edge.types.RemoveShareAccessControlEntry]`) – List of access control entries to remove

**remove\_screened\_file\_types**(*name, extensions*)

Remove extensions from the share's current list of blocked file extensions

**Parameters**

- **name** (`str`) – The share name
- **extensions** (`list[str]`) – List of file extensions to remove

**remove\_trusted\_nfs\_clients**(*name, trusted\_nfs\_clients*)

Remove one or more trusted NFS client entries from an existing share.

**Parameters**

- **name** (`str`) – The share name
- **trusted\_nfs\_clients** (`list[cterasdk.edge.types.RemoveNFSv3AccessControlEntry]`) – Trusted NFS v3 clients

**set\_access\_type**(*name, access*)

Set the network share Windows File Sharing authentication mode

**Parameters**

- **name** (`str`) – The share name
- **access** (`cterasdk.edge.enum.Acl`) – The Windows File Sharing authentication mode

**set\_acl**(*name, acl*)

Set a network share's access control entries.

**Parameters**

- **name** (`str`) – The share name



- **acl** (*list[cterasdk.edge.types.ShareAccessControlEntry]*) – List of access control entries

**Warning:** this method will override the existing access control entries

**set\_screened\_file\_types**(*name, extensions*)

Set the share's current list of blocked file extensions (override the current list)

**Parameters**

- **name** (*str*) – The share name
- **extensions** (*list[str]*) – List of file extensions to block

**set\_share\_winacls**(*name*)

Set a network share to use Windows ACL Emulation Mode

**Parameters** **name** (*str*) – The share name

**set\_trusted\_nfs\_clients**(*name, trusted\_nfs\_clients*)

Set a network share's trusted NFS client entries.

**Parameters**

- **name** (*str*) – The share name
- **trusted\_nfs\_clients** (*list[cterasdk.edge.types.NFSv3AccessControlEntry]*) – Trusted NFS v3 clients

**Warning:** this method will override the existing access control entries

### cterasdk.edge.shell module

**class** cterask.edge.shell.**Shell**(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Shell command

**run\_command**(*shell\_command, wait=True*)

Execute a shell command on the gateway

**Parameters**

- **shell\_command** (*str*) – The shell command to execute
- **wait** (*bool, optional*) – Wait for the command to execute, defaults to True

**Returns** The command result, or the task url path when wait equals False

**cterasdk.edge.smb module****class** `cterasdk.edge.smb.SMB(gateway)`Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway SMB configuration APIs

**disable()**

Disable SMB

**disable\_abe()**

Disable ABE

**enable()**

Enable SMB

**enable\_abe()**

Enable ABE

**get\_configuration()**

Get current SMB Configuration

**Return** `cterasdk.common.object.Object` SMB configuration**modify**(*packet\_signing=None, idle\_disconnect\_time=None, compatibility\_mode=None, unix\_extensions=None, abe\_enabled=None, min\_client\_protocol=None, max\_client\_protocol=None, min\_server\_protocol=None, max\_server\_protocol=None*)

Modify the current SMB Configuration. Parameters that are not passed will not be affected

**Parameters**

- **packet\_signing, optional** (`cterasdk.edge.enum.CIFSPacketSigning`) – Packet signing type
- **idle\_disconnect\_time** (*int, optional*) – Client idle disconnect timeout
- **compatibility\_mode** (*bool, optional*) – Enable/Disable compatibility mode
- **unix\_extensions** (*bool, optional*) – Enable/Disable unix extensions
- **abe\_enabled** (*bool, optional*) – Enable/Disable ABE
- **min\_client\_protocol** (`cterasdk.edge.enum.SMBProtocol, optional`) – Minimum client protocol version
- **max\_client\_protocol** (`cterasdk.edge.enum.SMBProtocol, optional`) – Maximum client protocol version
- **min\_server\_protocol** (`cterasdk.edge.enum.SMBProtocol, optional`) – Minimum server protocol version
- **max\_server\_protocol** (`cterasdk.edge.enum.SMBProtocol, optional`) – Maximum server protocol version

**restart()****set\_packet\_signing**(*packet\_signing*)

Set Packet signing

**Parameters** **packet\_signing** (`cterasdk.edge.enum.CIFSPacketSigning`) – Packet signing type

**cterasdk.edge.support module**

```
class cterasdk.edge.support.DebugLevel
    Bases: object
    aapi = 'aapi'
    alert = 'alert'
    apps = 'apps'
    auth = 'auth'
    av = 'av'
    backup = 'backup'
    caching = 'caching'
    cbck = 'cbck'
    cloud_extender = 'cloud_extender'
    collaboration = 'collaboration'
    cttp = 'cttp'
    cttp_data = 'cttp_data'
    db = 'db'
    debug = 'debug'
    dns = 'dns'
    error = 'error'
    error_abort = 'error_abort'
    evictor = 'evictor'
    evictor_verbose = 'evictor_verbose'
    files = 'files'
    http = 'http'
    index = 'index'
    info = 'info'
    license = 'license'
    none = 'none'
    ntp = 'ntp'
    process = 'process'
    rsync = 'rsync'
    samba = 'samba'
    storage = 'storage'
    upload = 'upload'
    warning = 'warning'
```

**class** `cterasdk.edge.support.Support`(*gateway*)  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Gateway Support APIs  
**get\_support\_report**()  
Download support report  
**set\_debug\_level**(\**levels*)  
Set the debug level

### **cterasdk.edge.sync module**

**class** `cterasdk.edge.sync.CloudSyncBandwidthThrottling`(*gateway*)  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Edge Filer Cloud Sync Bandwidth Throttling APIs  
**get\_policy**()  
Get the bandwidth throttling policy  
**Returns** a list of bandwidth throttling rules  
**Return type** `list[cterasdk.common.types.ThrottlingRule]`  
**set\_policy**(*rules*)  
Set the bandwidth throttling policy  
**Parameters** **rules** (`list[cterasdk.common.types.ThrottlingRule]`) – List of bandwidth throttling rules

**class** `cterasdk.edge.sync.Sync`(*portal*)  
Bases: `cterasdk.edge.base_command.BaseCommand`  
Edge Filer Cloud Sync APIs  
**Variables** **throttling** (`cterasdk.edge.sync.CloudSyncBandwidthThrottling`) – Object holding the Edge Filer's bandwidth throttling APIs  
**exclude\_files**(*extensions=None, filenames=None, paths=None, custom\_exclusion\_rules=None*)  
Exclude files from Cloud Sync. This method will override any existing file exclusion rules Use `cterasdk.common.types.FileFilterBuilder()` to build custom file exclusion rules`

#### **Parameters**

- **extensions** (`list[str]`) – List of file extensions
- **filenames** (`list[str]`) – List of file names
- **paths** (`list[str]`) – List of file paths
- **rules** (`list[cterasdk.common.types.FilterBackupSet]`) – Set of custom exclusion rules

**get\_linux\_avoid\_using\_fanotify**()  
**get\_status**()  
Retrieve the Cloud Sync status  
**is\_disabled**()  
Check if Cloud Sync is disabled  
**is\_enabled**()  
Check if Cloud Sync is enabled

**refresh()**

Refresh Cloud Folders

**remove\_file\_exclusion\_rules()**

Remove previously configured sync exclusion rules

**set\_linux\_avoid\_using\_fanotify(*avoid*)**

**suspend(*wait=True*)**

Suspend Cloud Sync

**Parameters** *wait* (*bool*) – Wait for synchronization to stop

**unsuspend()**

Unsuspend Cloud Sync

### cterasdk.edge.syslog module

**class** `cterasdk.edge.syslog.Syslog(gateway)`

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Syslog configuration APIs

**disable()**

Disable Syslog

**enable(*server, port=514, proto='UDP', min\_severity='info'*)**

Enable Syslog

#### Parameters

- **server** (*str*) – Server address to send syslog logs
- **port** (*int, optional*) – Syslog server communication port, defaults to 514
- **proto** (`cterasdk.edge.enum.IPProtocol`, *optional*) – Syslog server communication protocol, defaults to `cterasdk.edge.enum.IPProtocol.UDP`
- **min\_severity** (`cterasdk.edge.enum.Severity`, *optional*) – Minimal log severity to fetch, defaults to `cterasdk.edge.enum.Severity.INFO`

**get\_configuration()**

**modify(*server=None, port=None, proto=None, min\_severity=None*)**

Modify current Syslog configuration. Only configurations that are not None will be changed. Syslog must be enabled

#### Parameters

- **server** (*str, optional*) – Server address to send syslog logs
- **port** (*int, optional*) – Syslog server communication port
- **proto** (`cterasdk.edge.enum.IPProtocol`, *optional*) – Syslog server communication protocol
- **min\_severity** (`cterasdk.edge.enum.Severity`, *optional*) – Minimal log severity to fetch

### cterasdk.edge.taskmgr module

**class** cterasdk.edge.taskmgr.**Task**(CTERAHost, ref, retries=10, seconds=1)

Bases: cterasdk.lib.task\_manager\_base.TaskBase

**get\_task\_status**()

**class** cterasdk.edge.taskmgr.**Tasks**(gateway)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Background Task APIs

**by\_name**(name)

Get background tasks by name

**Parameters** **name** (str) – Task name

**running**()

Get all running background tasks

**status**(ref)

Get background task status

**Parameters** **ref** (str) – Task reference

**wait**(ref, retries=100, seconds=1)

Wait for background task to complete

**Parameters**

- **ref** (str) – Task reference
- **retries** (int, optional) – Number of retries when sampling the task status, defaults to 100
- **seconds** (int, optional) – Number of seconds to wait between retries, defaults to 1

### cterasdk.edge.telnet module

**class** cterasdk.edge.telnet.**Telnet**(gateway)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Telnet configuration APIs

**disable**()

Disable Telnet

**enable**(code)

Enable Telnet

### cterasdk.edge.timezone module

**class** cterasdk.edge.timezone.**Timezone**(gateway)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Timezone configuration

**get\_timezone**()

Get the timezone of the gateway

**Return str** The timezone of the gateway

**set\_timezone**(*timezone*)

Set Timezone

**Parameters** **timezone** (*str*) – New timezone to set

### cterasdk.edge.types module

**class** cterasdk.edge.types.**AccessControlEntryValidator**

Bases: object

**static validate\_permission**(*permission*)

**class** cterasdk.edge.types.**NFSv3AccessControlEntry**(*address, netmask, perm*)

Bases: object

NFS v3 export access control entry :ivar str address: IP address, hostname or fully qualified domain name of client machine :ivar str netmask: Subnet mask :ivar cterasdk.edge.enum.FileAccessMode perm: File access permission

**property address**

**static from\_server\_object**(*server\_object*)

**property netmask**

**property perm**

**to\_server\_object**()

**class** cterasdk.edge.types.**RemoveNFSv3AccessControlEntry**(*address, netmask*)

Bases: object

Object holding address and netmask for NFS v3 export access control entry :ivar str address: IP address, hostname or fully qualified domain name of client machine :ivar str netmask: Subnet mask

**property address**

**property netmask**

**class** cterasdk.edge.types.**RemoveShareAccessControlEntry**(*principal\_type, name*)

Bases: [cterasdk.edge.types.UserGroupEntry](#)

Object holding share access control principal type and name

#### Variables

- **principal\_type** ([cterasdk.edge.enum.PrincipalType](#)) – Principal type of the ACL
- **name** (*str*) – The name of the user or group

**class** cterasdk.edge.types.**ShareAccessControlEntry**(*principal\_type, name, perm*)

Bases: object

Share access control entry for filer shares

#### Variables

- **principal\_type** ([cterasdk.edge.enum.PrincipalType](#)) – Principal type of the ACL
- **name** (*str*) – The name of the user or group
- **perm** ([cterasdk.edge.enum.FileAccessMode](#)) – The file access permission

**static from\_server\_object**(*server\_object*)

**property name**

**property perm**

**property principal\_type**

**to\_server\_object()**

**class** cterasdk.edge.types.**TCPConnectResult**(*host, port, is\_open*)

Bases: tuple

Tuple holding the host and port to connect over TCP

**property host**

The ip address, hostname or fully qualified domain name of the host

**property is\_open**

Boolean, indicating whether a TCP connection can be successfully established to the target host over the specified port

**property port**

The port number

**class** cterasdk.edge.types.**TCPService**(*host, port*)

Bases: tuple

Tuple holding the host and port to connect over TCP

**property host**

The ip address, hostname or fully qualified domain name of the host

**property port**

The port number

**class** cterasdk.edge.types.**UserGroupEntry**(*principal\_type, name*)

Bases: object

User or Group Entry

**Variables**

- **principal\_type** (cterasdk.edge.enum.PrincipalType) – Principal type of the ACL
- **name** (str) – The name of the user or group

**static from\_server\_object**(*server\_object*)

**property principal\_type**

**to\_server\_object()**

### cterasdk.edge.uri module

cterasdk.edge.uri.**api**(*Gateway*)

cterasdk.edge.uri.**files**(*Gateway*)

cterasdk.edge.uri.**local**(*baseurl*)

cterasdk.edge.uri.**remote**(*baseurl, tenant, device*)

cterasdk.edge.uri.**remote\_access**(*baseurl, device*)



**cterasdk.edge.users module**

**class** `cterasdk.edge.users.Users`(*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Users configuration APIs

**add**(*username, password, full\_name=None, email=None, uid=None*)

Add a user of the Gateway

**Parameters**

- **username** (*str*) – User name for the new user
- **password** (*str*) – Password for the new user
- **full\_name** (*str, optional*) – The full name of the new user, defaults to None
- **email** (*str, optional*) – E-mail address of the new user, defaults to None
- **uid** (*str, optional*) – The uid of the new user, defaults to None

**add\_first\_user**(*username, password, email=""*)

Add the first user of the Gateway and login

**Parameters**

- **username** (*str*) – User name for the new user
- **password** (*str*) – Password for the new user
- **email** (*str, optional*) – E-mail address of the new user, defaults to an empty string

**delete**(*username*)

Delete an existing user

**Parameters** **username** (*str*) – User name of the user to delete

**get**(*name=None*)

Get User. If a user name was not passed as an argument, a list of all local users will be retrieved

**Parameters** **name** (*str, optional*) – Name of the user

**modify**(*username, password=None, full\_name=None, email=None, uid=None*)

Modify an existing user of the Gateway

**Parameters**

- **username** (*str*) – User name to modify
- **password** (*str, optional*) – New password, defaults to None
- **full\_name** (*str, optional*) – The full name of the user, defaults to None
- **email** (*str, optional*) – E-mail address of the user, defaults to None
- **uid** (*str, optional*) – The uid of the user, defaults to None

## cterasdk.edge.volumes module

**class** cterasdk.edge.volumes.Volumes(*gateway*)

Bases: *cterasdk.edge.base\_command.BaseCommand*

Gateway Volumes configuration APIs

**add**(*name*, *size=None*, *filesystem='xfs'*, *device=None*, *passphrase=None*)

Add a new volume to the gateway

### Parameters

- **name** (*str*) – Name of the new volume
- **size** (*int, optional*) – Size of the new volume, defaults to the device's size
- **filesystem** (*str, optional*) – Filesystem to use, defaults to xfs
- **device** (*str, optional*) – Name of the device to use for the new volume, can be left as None if there the gateway has only one
- **passphrase** (*str, optional*) – Passphrase for the volume

**Returns** Gateway response

**delete**(*name*)

Delete a volume

**Parameters** **name** (*str*) – Name of the volume to delete

**delete\_all**()

Delete all volumes

**get**(*name=None*)

Get Volume. If a volume name was not passed as an argument, a list of all storage volumes will be retrieved  
:param *str, optional name*: Name of the volume

**modify**(*name*, *size=None*)

Modify an existing volume

**Parameters** **size** (*int, optional*) – New size of the volume, if not set, the size will not change

**Returns** Gateway response

## 3.1.6 cterasdk.lib package

### 3.1.6.1 Submodules

#### cterasdk.lib.cmd module

**class** cterasdk.lib.cmd.Command(*cmd*, *\*args*)

Bases: object

**cterasdk.lib.consent module**

`cterasdk.lib.consent.ask(question)`

**cterasdk.lib.filesystem module**

**class** `cterasdk.lib.filesystem.FileSystem`

Bases: `object`

**static** `compute_zip_file_name(cloud_directory, files)`

**copyfile**(*src, dst*)

**static** `exists(filepath)`

**static** `expanduser(path)`

`get_dirpath()`

**static** `get_local_file_info(local_file)`

**static** `instance()`

**static** `join(*paths)`

`rename(dirpath, src, dst)`

`save(dirpath, filename, handle)`

**static** `split_file_directory(path)`

**static** `split_file_directory_with_defaults(path=None, default_filename=None)`

Compute the destination file path.

**Parameters**

- **path** (*str*) – A path to a file or a folder
- **default\_filename** (*str*) – The default file name to use unless *path* argument specifies a file path

**Returns** A tuple including the destination directory and filename

**Return type** (string, string)

`validate_directory(dirpath)`

**static** `version(filename, version)`

**static** `write(filepath, handle)`

**cterasdk.lib.file\_access\_base module**

**class** `cterasdk.lib.file_access_base.FileAccessBase(ctera_host)`

Bases: `abc.ABC`

**download**(*path, destination=None*)

**download\_as\_zip**(*cloud\_directory, files, destination=None*)

**upload**(*local\_file, dest\_path*)

### cterasdk.lib.iterator module

```
class cterasdk.lib.iterator.Iterator(function, param)  
    Bases: object  
    Objects Iterator
```

### cterasdk.lib.platform module

```
class cterasdk.lib.platform.Platform  
    Bases: object  
    arch()  
    static instance()  
    os()  
    python_version()
```

### cterasdk.lib.registry module

```
class cterasdk.lib.registry.Registry  
    Bases: object  
    get(key)  
    static instance()  
    register(key, value)  
    remove(key)
```

### cterasdk.lib.session\_base module

```
class cterasdk.lib.session_base.SessionBase(host)  
    Bases: cterasdk.common.object.Object  
    property active  
    authenticated()  
    initializing()  
    is_local_auth()  
    start_local_session(ctera_host)  
    tenant()  
    terminate()  
    whoami()  
class cterasdk.lib.session_base.SessionStatus  
    Bases: object  
    Active = 'Active'  
    Inactive = 'Inactive'
```

```
    Initializing = 'Initializing'
```

```
class cterasdk.lib.session_base.SessionUser(name, tenant=None, role=None)
    Bases: cterasdk.common.object.Object
```

### cterasdk.lib.tempfile module

```
class cterasdk.lib.tempfile.TempfileServices
    Bases: object
    static mkdir()
    static mkfile(prefix, suffix)
    static rmdir()
```

### cterasdk.lib.tracker module

```
exception cterasdk.lib.tracker.ErrorStatus(status)
    Bases: cterasdk.exception.CTERAException

class cterasdk.lib.tracker.StatusTracker(CTERAHost, ref, success, progress, transient, failure, retries,
                                          seconds)
    Bases: object
    failed()
    increment()
    resolve()
    running()
    successful()
    track()

cterasdk.lib.tracker.track(CTERAHost, ref, success, progress, transient, failure, retries=300, seconds=1)
```

### cterasdk.lib.version module

```
class cterasdk.lib.version.Version
    Bases: object
    as_header()
    static instance()
```

## 3.1.7 cterasdk.object package

### 3.1.7.1 Submodules

#### cterasdk.object.Agent module

**class** `cterasdk.object.Agent.Agent`(*host*, *port=80*, *https=False*, *Portal=None*)

Bases: `cterasdk.client.host.CTERAHost`

Main class operating on a Agent

##### Variables

- **backup** (`cterasdk.edge.backup.Backup`) – Object holding the Agent Backup APIs
- **cli** (`cterasdk.edge.cli.CLI`) – Object holding the Agent CLI APIs
- **logs** (`cterasdk.edge.logs.Logs`) – Object holding the Agent Logs APIs
- **services** (`cterasdk.edge.services.Services`) – Object holding the Agent Services APIs
- **support** (`cterasdk.edge.support.Support`) – Object holding the Agent Support APIs
- **sync** (`cterasdk.edge.sync.Sync`) – Object holding the Agent Sync APIs

property `base_api_url`

#### cterasdk.object.Gateway module

**class** `cterasdk.object.Gateway.Gateway`(*host*, *port=None*, *https=False*, *Portal=None*)

Bases: `cterasdk.client.host.CTERAHost`

Main class operating on a Gateway

##### Variables

- **config** (`cterasdk.edge.config.Config`) – Object holding the Gateway Configuration APIs
- **network** (`cterasdk.edge.network.Network`) – Object holding the Gateway Network APIs
- **licenses** (`cterasdk.edge.licenses.Licenses`) – Object holding the Gateway Licenses APIs
- **services** (`cterasdk.edge.services.Services`) – Object holding the Gateway Services APIs
- **directoryservice** (`cterasdk.edge.directoryservice.DirectoryService`) – Object holding the Gateway Active Directory APIs
- **telnet** (`cterasdk.edge.telnet.Telnet`) – Object holding the Gateway Telnet APIs
- **syslog** (`cterasdk.edge.syslog.Syslog`) – Object holding the Gateway Syslog APIs
- **tasks** (`cterasdk.edge.taskmgr.Tasks`) – Object holding the Gateway Background Tasks APIs
- **audit** (`cterasdk.edge.audit.Audit`) – Object holding the Gateway Audit APIs
- **mail** (`cterasdk.edge.mail.Mail`) – Object holding the Gateway Mail APIs

- **backup** (`cterasdk.edge.backup.Backup`) – Object holding the Gateway Backup APIs
- **sync** (`cterasdk.edge.sync.Sync`) – Object holding the Gateway Sync APIs
- **cache** (`cterasdk.edge.cache.Cache`) – Object holding the Gateway Cache APIs
- **snmp** (`cterasdk.edge.snmp.SNMP`) – Object holding the Gateway SNMP APIs
- **ssl** (`cterasdk.edge.ssl.SSL`) – Object holding the Gateway SSL APIs
- **ssh** (`cterasdk.edge.ssh.SSH`) – Object holding the Gateway SSH APIs
- **power** (`cterasdk.edge.power.Power`) – Object holding the Gateway Power APIs
- **users** (`cterasdk.edge.users.Users`) – Object holding the Gateway Users APIs
- **groups** (`cterasdk.edge.groups.Groups`) – Object holding the Gateway Groups APIs
- **drive** (`cterasdk.edge.drive.Drive`) – Object holding the Gateway Drive APIs
- **volumes** (`cterasdk.edge.volumes.Volumes`) – Object holding the Gateway Volumes APIs
- **array** (`cterasdk.edge.array.Array`) – Object holding the Gateway Array APIs
- **shares** (`cterasdk.edge.shares.Shares`) – Object holding the Gateway Shares APIs
- **smb** (`cterasdk.edge.smb.SMB`) – Object holding the Gateway SMB APIs
- **aio** (`cterasdk.edge.aio.AIO`) – Object holding the Gateway AIO APIs
- **ftp** (`cterasdk.edge.ftp.FTP`) – Object holding the Gateway FTP APIs
- **afp** (`cterasdk.edge.afp.AFP`) – Object holding the Gateway AFP APIs
- **nfs** (`cterasdk.edge.nfs.NFS`) – Object holding the Gateway NFS APIs
- **rsync** (`cterasdk.edge.rsync.RSync`) – Object holding the Gateway RSync APIs
- **timezone** (`cterasdk.edge.timezone.Timezone`) – Object holding the Gateway Timezone APIs
- **logs** (`cterasdk.edge.logs.Logs`) – Object holding the Gateway Logs APIs
- **ntp** (`cterasdk.edge.ntp.NTP`) – Object holding the Gateway NTP APIs
- **shell** (`cterasdk.edge.shell.Shell`) – Object holding the Gateway Shell APIs
- **cli** (`cterasdk.edge.cli.CLI`) – Object holding the Gateway CLI APIs
- **support** (`cterasdk.edge.support.Support`) – Object holding the Gateway Support APIs
- **files** (`cterasdk.edge.files.FileBrowser`) – Object holding the Gateway File Browsing APIs
- **firmware** (`cterasdk.edge.firmware.Firmware`) – Object holding the Gateway Firmware APIs

**property** `base_api_url`

**property** `base_file_url`

**property** `initialized`

**static** `make_local_files_dir(full_path)`

**query**(`path, key, value`)

**remote\_access**()

**rm**(*path*)

**show\_query**(*path, key, value*)

**test**()

Verification check to ensure the target host is a Gateway.

### **cterasdk.object.Portal module**

**class** `cterasdk.object.Portal.GlobalAdmin`(*host, port=None, https=True*)

Bases: `cterasdk.object.Portal.Portal`

Main class for Global Admin operations on a Portal

#### **Variables**

- **portals** (`cterasdk.core.portals.Portals`) – Object holding the Portals Management APIs
- **servers** (`cterasdk.core.servers.Servers`) – Object holding the Servers Management APIs
- **setup** (`cterasdk.core.setup.Setup`) – Object holding the Portal setup APIs
- **ssl** (`cterasdk.core.ssl.SSL`) – Object holding the Portal SSL Certificate APIs
- **startup** (`cterasdk.core.startup.Startup`) – Object holding the Portal startup APIs
- **syslog** (`cterasdk.core.syslog.Syslog`) – Object holding the Portal syslog APIs
- **antivirus** (`cterasdk.core.antivirus.Antivirus`) – Object holding the Portal Antivirus APIs
- **buckets** (`cterasdk.core.buckets.Buckets`) – Object holding the Portal Storage Node APIs

**property** `backups_base_path`

**property** `cloud_drive_base_path`

**property** `context`

**class** `cterasdk.object.Portal.Portal`(*host, port, https*)

Bases: `cterasdk.client.host.CTERAHost`

Parent class for communicating with the Portal through either GlobalAdmin or ServicesPortal

#### **Variables**

- **users** (`cterasdk.core.users.Users`) – Object holding the Portal user APIs
- **plans** (`cterasdk.core.plans.Plans`) – Object holding the Plan APIs
- **reports** (`cterasdk.core.reports.Reports`) – Object holding the Portal reports APIs
- **devices** (`cterasdk.core.devices.Devices`) – Object holding the Portal devices APIs
- **directoryservice** (`cterasdk.core.directoryservice.DirectoryService`) – Object holding the Portal Active Directory Service APIs
- **zones** (`cterasdk.core.zones.Zones`) – Object holding the Portal zones APIs
- **activation** (`cterasdk.core.activation.Activation`) – Object holding the Portal activation APIs
- **logs** (`cterasdk.core.logs.Logs`) – Object holding the Portal logs APIs



- **cloudfs** (`cterasdk.core.cloudfs.CloudFS`) – Object holding the Portal CloudFS APIs
- **settings** (`cterasdk.core.settings.Settings`) – Object holding the Portal Settings APIs
- **tasks** (`cterasdk.core.taskmgr.Tasks`) – Object holding the Portal Background Tasks APIs
- **templates** (`cterasdk.core.templates.Templates`) – Object holding the Portal Configuration Templates APIs
- **firmwares** (`cterasdk.core.firmwares.Firmwares`) – Object holding the Portal Firmware Repository APIs
- **files** (`cterasdk.core.files.browser.FileBrowser`) – Object holding the Portal File Browsing APIs

**property backups\_base\_path**

**property base\_api\_url**

**property base\_file\_url**

**property base\_portal\_url**

**property cloud\_drive\_base\_path**

**property context**

**iterator**(*path, param*)

**public\_info**()

Obtain the Portal's public info.

**put**(*path, value, use\_file\_url=False*)

Update a schema object or attribute.

**query**(*path, param*)

**show\_query**(*path, param*)

**test**()

Verification check to ensure the target host is a Portal.

**class** `cterasdk.object.Portal.ServicesPortal`(*host, port=None, https=True*)

Bases: `cterasdk.object.Portal.Portal`

Main class for Service operations on a Portal

**property backups\_base\_path**

**property cloud\_drive\_base\_path**

**property context**

## 3.1.8 cterasdk.transcript package

### 3.1.8.1 Submodules

#### cterasdk.transcript.transcribe module

```
class cterasdk.transcript.transcribe.Transcribe
    Bases: object
    COMMENT = '<!--TEMPLATE-->'
    RECORDED_HEADERS = ['content-type', 'cookie']
    SECTION_END = '</div>'
    SECTION_START = '<div style="border: 1px solid #EBECF0; margin-bottom: 10px;">'
    transcribe(request, response=None)
cterasdk.transcript.transcribe.transcribe(request, response=None)
```

## 3.2 Submodules

### 3.2.1 cterasdk.config module

```
class cterasdk.config.Logging
    Bases: object
    static df()
    static disable()
    static enable()
    static fmt()
    static get()
    static setLevel(level)
```

### 3.2.2 cterasdk.exception module

```
exception cterasdk.exception.CTERAClientException(message=None, instance=None, **kwargs)
    Bases: cterasdk.exception.CTERAException
exception cterasdk.exception.CTERAConnectionError(message, instance, host, port, protocol, **kwargs)
    Bases: cterasdk.exception.CTERAException
exception cterasdk.exception.CTERAException(message=None, instance=None, **kwargs)
    Bases: Exception
    join(instance=None)
    put(**kwargs)
exception cterasdk.exception.ConnectionTimeout(message, seconds, **kwargs)
    Bases: cterasdk.exception.CTERAException
```

**exception** `cterasdk.exception.ConsentException`  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.ExhaustedException`(*retries, timeout*)  
Bases: `cterasdk.exception.ConnectionTimeout`

**exception** `cterasdk.exception.FileSystemException`(*message=None, instance=None, \*\*kwargs*)  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.HostUnreachable`(*instance, host, port, protocol*)  
Bases: `cterasdk.exception.CTERAConnectionError`

**exception** `cterasdk.exception.InputError`(*message, expression, options*)  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.LocalDirectoryNotFound`(*path*)  
Bases: `cterasdk.exception.FileSystemException`

**exception** `cterasdk.exception.LocalFileNotFound`(*path*)  
Bases: `cterasdk.exception.FileSystemException`

**exception** `cterasdk.exception.LocalPathNotFound`(*path*)  
Bases: `cterasdk.exception.FileSystemException`

**exception** `cterasdk.exception.ObjectNotFoundException`(*message, object\_ref, \*\*kwargs*)  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.PythonVersionException`(*version*)  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.RemoteDirectoryNotFound`(*path*)  
Bases: `cterasdk.exception.RemoteFileSystemException`

**exception** `cterasdk.exception.RemoteFileSystemException`(*message=None, instance=None, \*\*kwargs*)  
Bases: `cterasdk.exception.CTERAException`

**exception** `cterasdk.exception.RenameException`(*dirpath, src, dst*)  
Bases: `cterasdk.exception.FileSystemException`

**exception** `cterasdk.exception.SSLException`(*host, port, reason*)  
Bases: `cterasdk.exception.CTERAConnectionError`



## INDICES AND TABLES

- genindex
- modindex
- search



## HELP US IMPROVE THE DOCS <3

If you'd like to contribute an improvement to the site, its source is available on GitHub. Simply fork the repository and submit a pull request. Thank you!





## PYTHON MODULE INDEX

### C

- cterasdk, 79
- cterasdk.client, 79
  - cterasdk.client.cteraclient, 79
  - cterasdk.client.host, 80
  - cterasdk.client.http, 81
  - cterasdk.client.ssl, 83
- cterasdk.common, 83
  - cterasdk.common.datetime\_utils, 83
  - cterasdk.common.item, 83
  - cterasdk.common.object, 83
  - cterasdk.common.types, 84
- cterasdk.config, 182
- cterasdk.convert, 88
  - cterasdk.convert.exception, 88
  - cterasdk.convert.format, 88
  - cterasdk.convert.parse, 88
  - cterasdk.convert.xml\_types, 89
- cterasdk.core, 89
  - cterasdk.core.activation, 95
  - cterasdk.core.antivirus, 95
  - cterasdk.core.base\_command, 96
  - cterasdk.core.buckets, 96
  - cterasdk.core.cloudfs, 97
  - cterasdk.core.connection, 99
  - cterasdk.core.decorator, 99
  - cterasdk.core.devices, 99
  - cterasdk.core.directoryservice, 101
  - cterasdk.core.enum, 103
  - cterasdk.core.files, 89
    - cterasdk.core.files.browser, 89
    - cterasdk.core.files.collaboration, 92
    - cterasdk.core.files.common, 93
    - cterasdk.core.files.cp, 93
    - cterasdk.core.files.directory, 93
    - cterasdk.core.files.file\_access, 93
    - cterasdk.core.files.ln, 94
    - cterasdk.core.files.ls, 94
    - cterasdk.core.files.mv, 94
    - cterasdk.core.files.path, 94
    - cterasdk.core.files.recover, 94
    - cterasdk.core.files.rename, 94
    - cterasdk.core.files.rm, 95
  - cterasdk.core.login, 115
  - cterasdk.core.logs, 115
  - cterasdk.core.plans, 120
  - cterasdk.core.portals, 116
  - cterasdk.core.query, 118
  - cterasdk.core.remote, 121
  - cterasdk.core.reports, 119
  - cterasdk.core.servers, 121
  - cterasdk.core.session, 122
  - cterasdk.core.setup, 122
  - cterasdk.core.startup, 123
  - cterasdk.core.syslog, 123
  - cterasdk.core.taskmgr, 124
  - cterasdk.core.templates, 125
  - cterasdk.core.types, 127
  - cterasdk.core.users, 131
  - cterasdk.core.zones, 132
- cterasdk.edge, 133
  - cterasdk.edge.afp, 136
  - cterasdk.edge.aio, 136
  - cterasdk.edge.array, 136
  - cterasdk.edge.audit, 137
  - cterasdk.edge.backup, 137
  - cterasdk.edge.base\_command, 139
  - cterasdk.edge.cache, 139
  - cterasdk.edge.cli, 140
  - cterasdk.edge.config, 140
  - cterasdk.edge.connection, 141
  - cterasdk.edge.decorator, 141
  - cterasdk.edge.directoryservice, 141
  - cterasdk.edge.drive, 142
  - cterasdk.edge.enum, 143
  - cterasdk.edge.files, 133
    - cterasdk.edge.files.browser, 133
    - cterasdk.edge.files.copy, 135
    - cterasdk.edge.files.file\_access, 135
    - cterasdk.edge.files.mkdir, 135
    - cterasdk.edge.files.move, 135
    - cterasdk.edge.files.path, 135
    - cterasdk.edge.files.rm, 135
  - cterasdk.edge.firmware, 153

- cterasdk.edge.ftp, 152
- cterasdk.edge.groups, 153
- cterasdk.edge.licenses, 154
- cterasdk.edge.login, 154
- cterasdk.edge.logs, 154
- cterasdk.edge.mail, 155
- cterasdk.edge.network, 155
- cterasdk.edge.nfs, 157
- cterasdk.edge.ntp, 157
- cterasdk.edge.power, 159
- cterasdk.edge.query, 159
- cterasdk.edge.remote, 160
- cterasdk.edge.rsync, 160
- cterasdk.edge.services, 160
- cterasdk.edge.session, 161
- cterasdk.edge.shares, 162
- cterasdk.edge.shell, 165
- cterasdk.edge.smb, 166
- cterasdk.edge.ssh, 158
- cterasdk.edge.ssl, 158
- cterasdk.edge.support, 167
- cterasdk.edge.sync, 168
- cterasdk.edge.syslog, 169
- cterasdk.edge.taskmgr, 170
- cterasdk.edge.telnet, 170
- cterasdk.edge.timezone, 170
- cterasdk.edge.types, 171
- cterasdk.edge.uri, 172
- cterasdk.edge.users, 173
- cterasdk.edge.volumes, 174
- cterasdk.exception, 182
- cterasdk.lib, 174
  - cmd, 174
  - consent, 175
  - file\_access\_base, 175
  - filesystem, 175
  - iterator, 176
  - platform, 176
  - registry, 176
  - session\_base, 176
  - tempfile, 177
  - tracker, 177
  - version, 177
- cterasdk.object, 178
  - Agent, 178
  - Gateway, 178
  - Portal, 180
- cterasdk.transcript, 182
  - transcript.transcribe, 182

## A

- aapi (*cterasdk.edge.support.DebugLevel* attribute), 167
- Access (*cterasdk.core.enum.LogTopic* attribute), 108
- AccessControlEntry (class in *cterasdk.core.types*), 127
- AccessControlEntryValidator (class in *cterasdk.edge.types*), 171
- AccessControlRule (class in *cterasdk.core.types*), 127
- account (*cterasdk.core.types.AccessControlEntry* property), 127
- account\_type (*cterasdk.core.types.GroupAccount* property), 128
- account\_type (*cterasdk.core.types.PortalAccount* property), 129
- account\_type (*cterasdk.core.types.UserAccount* property), 130
- Acl (class in *cterasdk.edge.enum*), 143
- ActionResourcesParam (class in *cterasdk.core.files.common*), 93
- activate() (*cterasdk.edge.services.Services* method), 160
- Activation (class in *cterasdk.core.activation*), 95
- active (*cterasdk.lib.session\_base.SessionBase* property), 176
- Active (*cterasdk.lib.session\_base.SessionStatus* attribute), 176
- add() (*cterasdk.client.host.CTERAHost* method), 80
- add() (*cterasdk.core.antivirus.AntivirusServers* method), 95
- add() (*cterasdk.core.buckets.Buckets* method), 96
- add() (*cterasdk.core.files.common.ActionResourcesParam* method), 93
- add() (*cterasdk.core.plans.Plans* method), 120
- add() (*cterasdk.core.portals.Portals* method), 116
- add() (*cterasdk.core.templates.Templates* method), 125
- add() (*cterasdk.core.users.Users* method), 131
- add() (*cterasdk.core.zones.Zones* method), 132
- add() (*cterasdk.edge.array.Array* method), 136
- add() (*cterasdk.edge.licenses.LocalLicenses* method), 154
- add() (*cterasdk.edge.shares.Shares* method), 162
- add() (*cterasdk.edge.users.Users* method), 173
- add() (*cterasdk.edge.volumes.Volumes* method), 174
- add\_acl() (*cterasdk.edge.shares.Shares* method), 162
- add\_devices() (*cterasdk.core.zones.Zones* method), 132
- add\_first\_user() (*cterasdk.edge.users.Users* method), 173
- add\_folders() (*cterasdk.core.zones.Zones* method), 133
- add\_members() (*cterasdk.edge.groups.Groups* method), 153
- add\_screened\_file\_types() (*cterasdk.edge.shares.Shares* method), 162
- add\_share\_recipients() (*cterasdk.core.files.browser.CloudDrive* method), 89
- add\_share\_recipients() (in module *cterasdk.core.files.collaboration*), 92
- add\_trusted\_cert() (*cterasdk.client.ssl.CertificateServices* static method), 83
- add\_trusted\_nfs\_clients() (*cterasdk.edge.shares.Shares* method), 163
- addFilter() (*cterasdk.core.query.QueryParamBuilder* method), 118
- ADDomainIDMapping (class in *cterasdk.core.types*), 127
- address (*cterasdk.edge.types.NFSv3AccessControlEntry* property), 171
- address (*cterasdk.edge.types.RemoveNFSv3AccessControlEntry* property), 171
- admin (*cterasdk.core.enum.Context* attribute), 104
- Administration (*cterasdk.core.session.Session* attribute), 122
- Administrators (*cterasdk.edge.enum.LocalGroup* attribute), 146
- advanced\_mapping() (*cterasdk.edge.directoryservice.DirectoryService* method), 141
- AdvancedFilterRule (class in *cterasdk.common.types*), 84
- AFP (class in *cterasdk.edge.afp*), 136
- after() (*cterasdk.common.types.DateTimeCriteriaBuilder* method), 85
- after() (*cterasdk.core.query.FilterBuilder* method), 118
- after\_backup() (*cterasdk.core.types.TemplateScript*

- method), 130
- after\_logon() (cterasdk.core.types.TemplateScript method), 130
- AfterOperator (class in cterasdk.common.types), 84
- Agent (class in cterasdk.object.Agent), 178
- Agents (cterasdk.core.enum.DeviceType attribute), 105
- AGENTS (cterasdk.core.enum.EnvironmentVariables attribute), 106
- agents() (cterasdk.core.devices.Devices method), 99
- AIO (class in cterasdk.edge.aio), 136
- ALERT (cterasdk.core.enum.Severity attribute), 114
- ALERT (cterasdk.edge.enum.Severity attribute), 148
- alert (cterasdk.edge.support.DebugLevel attribute), 167
- All (cterasdk.core.enum.ListFilter attribute), 108
- All (cterasdk.core.enum.PlanRetention attribute), 110
- ALL (cterasdk.core.enum.PolicyType attribute), 111
- ALL\_FILES (cterasdk.edge.backup.BackupFiles attribute), 138
- allPortals() (cterasdk.core.query.QueryParamBuilder method), 118
- ALLUSERSPROFILE (cterasdk.core.enum.EnvironmentVariables attribute), 106
- AmazonS3 (class in cterasdk.core.types), 127
- Antivirus (class in cterasdk.core.antivirus), 95
- Antivirus (cterasdk.core.enum.ICAPServices attribute), 107
- AntivirusServers (class in cterasdk.core.antivirus), 95
- AntivirusType (class in cterasdk.core.enum), 103
- api() (in module cterasdk.edge.uri), 172
- APPDATA (cterasdk.core.enum.EnvironmentVariables attribute), 106
- Apple (cterasdk.core.enum.DirectoryServiceType attribute), 106
- ApplicationBackupSet (class in cterasdk.common.types), 84
- apply() (cterasdk.edge.licenses.Licenses method), 154
- apply\_changes() (cterasdk.core.portals.Portals method), 117
- apply\_changes() (cterasdk.core.templates.TemplateAutoAssignPolicy method), 125
- apply\_changes() (cterasdk.core.users.Users method), 131
- apps (cterasdk.edge.support.DebugLevel attribute), 167
- arch() (cterasdk.lib.platform.Platform method), 176
- Array (class in cterasdk.edge.array), 136
- as\_header() (cterasdk.lib.version.Version method), 177
- ask() (in module cterasdk.lib.consent), 175
- ATT (cterasdk.convert.xml\_types.XMLTypes attribute), 89
- Attached (cterasdk.edge.enum.BackupConfStatusID attribute), 144
- AttachEncrypted, 137
- Attaching (cterasdk.edge.enum.BackupConfStatusID attribute), 144
- Attaching (cterasdk.edge.enum.ServicesConnectionState attribute), 148
- AttachRC (class in cterasdk.edge.backup), 137
- Audit (class in cterasdk.edge.audit), 137
- Audit (cterasdk.core.enum.LogTopic attribute), 108
- AuditEvents (class in cterasdk.edge.enum), 143
- auth (cterasdk.edge.support.DebugLevel attribute), 167
- authenticated() (cterasdk.lib.session\_base.SessionBase method), 176
- authenticated() (in module cterasdk.client.host), 81
- authenticated() (in module cterasdk.edge.decorator), 141
- Authenticating (cterasdk.edge.enum.ServicesConnectionState attribute), 148
- av (cterasdk.edge.support.DebugLevel attribute), 167
- AverageBlockSize (cterasdk.core.enum.DeduplicationMethodType attribute), 104
- AWS (cterasdk.core.enum.BucketType attribute), 103
- Azure (cterasdk.core.enum.BucketType attribute), 103
- Azure (cterasdk.core.enum.LocationType attribute), 108
- AzureBlob (class in cterasdk.core.types), 127
- ## B
- Backup (class in cterasdk.edge.backup), 138
- backup (cterasdk.edge.support.DebugLevel attribute), 167
- BackupConfStatusID (class in cterasdk.edge.enum), 143
- BackupFiles (class in cterasdk.edge.backup), 138
- Backups (class in cterasdk.core.files.browser), 89
- backups\_base\_path (cterasdk.object.Portal.GlobalAdmin property), 180
- backups\_base\_path (cterasdk.object.Portal.Portal property), 181
- backups\_base\_path (cterasdk.object.Portal.ServicesPortal property), 181
- BackupScheduleBuilder (class in cterasdk.common.types), 84
- BackupSet (class in cterasdk.common.types), 85
- base\_api\_url (cterasdk.client.host.CTERAHost property), 80
- base\_api\_url (cterasdk.object.Agent.Agent property), 178
- base\_api\_url (cterasdk.object.Gateway.Gateway property), 179
- base\_api\_url (cterasdk.object.Portal.Portal property), 181
- base\_file\_url (cterasdk.client.host.CTERAHost property), 80
- base\_file\_url (cterasdk.object.Gateway.Gateway property), 179
- base\_file\_url (cterasdk.object.Portal.Portal property), 181

- base\_portal\_url (*cterasdk.object.Portal.Portal* property), 181
- BaseCommand (class in *cterasdk.core.base\_command*), 96
- BaseCommand (class in *cterasdk.edge.base\_command*), 139
- baseurl() (*cterasdk.client.host.NetworkHost* method), 81
- before() (*cterasdk.common.types.DateTimeCriteriaBuilder* method), 85
- before() (*cterasdk.core.query.FilterBuilder* method), 118
- before\_backup() (*cterasdk.core.types.TemplateScript* method), 130
- BeforeOperator (class in *cterasdk.common.types*), 85
- BeginsWithOperator (class in *cterasdk.common.types*), 85
- billing\_id() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128
- BillingId (*cterasdk.core.enum.PlanCriteria* attribute), 109
- block\_files() (*cterasdk.edge.shares.Shares* method), 163
- Boolean (*cterasdk.core.query.FilterType* attribute), 118
- BooleanRefFilter (*cterasdk.core.query.FilterType* attribute), 118
- Boot (class in *cterasdk.edge.power*), 159
- browse() (*cterasdk.core.portals.Portals* method), 117
- browse\_global\_admin() (*cterasdk.core.portals.Portals* method), 117
- Bucket (class in *cterasdk.core.types*), 127
- Buckets (class in *cterasdk.core.buckets*), 96
- BucketType (class in *cterasdk.core.enum*), 103
- build() (*cterasdk.common.types.CriteriaBuilder* method), 85
- build() (*cterasdk.common.types.ThrottlingRuleBuilder* method), 87
- build() (*cterasdk.common.types.TimeRange* method), 87
- build() (*cterasdk.core.query.QueryParamBuilder* method), 118
- build() (*cterasdk.edge.query.QueryParamBuilder* method), 159
- by\_name() (*cterasdk.core.devices.Devices* method), 99
- by\_name() (*cterasdk.core.plans.Plans* method), 120
- by\_name() (*cterasdk.core.templates.Templates* method), 125
- by\_name() (*cterasdk.edge.taskmgr.Tasks* method), 170
- C**
- C200 (*cterasdk.core.enum.DeviceType* attribute), 105
- C200\_ARM (*cterasdk.core.enum.Platform* attribute), 111
- C200\_Kirkwood (*cterasdk.core.enum.Platform* attribute), 111
- C200\_Orion (*cterasdk.core.enum.Platform* attribute), 111
- C400 (*cterasdk.core.enum.DeviceType* attribute), 105
- C400\_C800 (*cterasdk.core.enum.Platform* attribute), 111
- C800 (*cterasdk.core.enum.DeviceType* attribute), 105
- C800P (*cterasdk.core.enum.DeviceType* attribute), 105
- Cache (class in *cterasdk.edge.cache*), 139
- caching (*cterasdk.edge.support.DebugLevel* attribute), 167
- CachingGateway (*cterasdk.edge.enum.OperationMode* attribute), 146
- CatalogReadOnlyMode (*cterasdk.edge.enum.SyncStatus* attribute), 149
- cbck (*cterasdk.edge.support.DebugLevel* attribute), 167
- CertificateServices (class in *cterasdk.client.ssl*), 83
- ChangeOwner (*cterasdk.edge.enum.AuditEvents* attribute), 143
- ChangePermissions (*cterasdk.edge.enum.AuditEvents* attribute), 143
- CheckCodeInCorrect (*cterasdk.edge.backup.AttachRC* attribute), 137
- Checking (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- CheckingQuota (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- CIFSPacketSigning (class in *cterasdk.edge.enum*), 144
- CLASS (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89
- clear() (*cterasdk.edge.licenses.LocalLicenses* method), 154
- CLI (class in *cterasdk.edge.cli*), 140
- ClientSideCaching (class in *cterasdk.edge.enum*), 144
- ClocksOutOfSync, 138
- ClocksOutOfSync (*cterasdk.edge.backup.AttachRC* attribute), 137
- ClocksOutOfSync (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- ClocksOutOfSync (*cterasdk.edge.enum.SyncStatus* attribute), 150
- cloud\_drive\_base\_path (*cterasdk.object.Portal.GlobalAdmin* property), 180
- cloud\_drive\_base\_path (*cterasdk.object.Portal.Portal* property), 181
- cloud\_drive\_base\_path (*cterasdk.object.Portal.ServicesPortal* property), 181
- cloud\_extender (*cterasdk.edge.support.DebugLevel* attribute), 167
- CloudBackup (*cterasdk.core.enum.LogTopic* attribute), 108
- CloudDrive (class in *cterasdk.core.files.browser*), 89
- CloudFS (class in *cterasdk.core.cloudfs*), 97



- CloudFSFolderFindingHelper (class in *cterasdk.core.types*), 127
- CloudPlug (*cterasdk.core.enum.DeviceType* attribute), 105
- CloudSync (*cterasdk.core.enum.LogTopic* attribute), 108
- CloudSyncBandwidthThrottling (class in *cterasdk.edge.sync*), 168
- collaboration (*cterasdk.edge.support.DebugLevel* attribute), 167
- CollaboratorType (class in *cterasdk.core.enum*), 104
- Command (class in *cterasdk.lib.cmd*), 174
- Comment (*cterasdk.core.enum.PlanCriteria* attribute), 109
- COMMENT (*cterasdk.transcript.transcribe.Transcribe* attribute), 182
- comment() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128
- Company (*cterasdk.core.enum.PlanCriteria* attribute), 109
- company() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128
- COMPLETE (*cterasdk.edge.firmware.UploadTaskStatus* attribute), 153
- Completed (*cterasdk.core.enum.SetupWizardStatus* attribute), 113
- Completed (*cterasdk.edge.enum.TaskStatus* attribute), 150
- compute\_zip\_file\_name() (*cterasdk.lib.filesystem.FileSystem* static method), 175
- Config (class in *cterasdk.edge.config*), 140
- configure() (*cterasdk.edge.backup.Backup* method), 138
- Configuring (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- Connect (*cterasdk.core.enum.PlanItem* attribute), 110
- connect() (*cterasdk.core.directoryservice.DirectoryService* method), 101
- connect() (*cterasdk.edge.directoryservice.DirectoryService* method), 141
- connect() (*cterasdk.edge.services.Services* method), 160
- Connected (*cterasdk.edge.enum.ServicesConnectionState* attribute), 148
- Connected (*cterasdk.edge.enum.SyncStatus* attribute), 150
- connected() (*cterasdk.core.directoryservice.DirectoryService* method), 101
- connected() (*cterasdk.edge.directoryservice.DirectoryService* method), 141
- connected() (*cterasdk.edge.services.Services* method), 161
- Connecting (*cterasdk.edge.enum.ServicesConnectionState* attribute), 148
- ConnectingFolders (*cterasdk.edge.enum.SyncStatus* attribute), 150
- ConnectionFailed (*cterasdk.edge.enum.SyncStatus* attribute), 150
- ConnectionTimeout, 182
- ConsentException, 182
- contains() (*cterasdk.common.types.StringCriteriaBuilder* method), 86
- ContainsErrors (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- ContainsOperator (class in *cterasdk.common.types*), 85
- ContentType (class in *cterasdk.client.http*), 81
- Context (class in *cterasdk.core.enum*), 104
- context (*cterasdk.object.Portal.GlobalAdmin* property), 180
- context (*cterasdk.object.Portal.Portal* property), 181
- context (*cterasdk.object.Portal.ServicesPortal* property), 181
- convert() (*cterasdk.common.types.PolicyRuleConverter* static method), 86
- Converting (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- copy() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- copy() (*cterasdk.client.host.CTERAHost* method), 80
- copy() (*cterasdk.client.http.HTTPClient* method), 81
- copy() (*cterasdk.core.files.browser.FileBrowser* method), 91
- copy() (*cterasdk.edge.files.browser.FileBrowser* method), 133
- copy() (in module *cterasdk.core.files.cp*), 93
- copy() (in module *cterasdk.edge.files.copy*), 135
- copy\_multi() (*cterasdk.core.files.browser.FileBrowser* method), 91
- copy\_multi() (in module *cterasdk.core.files.cp*), 93
- copyfile() (*cterasdk.lib.filesystem.FileSystem* method), 175
- Corrupted (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- countLimit() (*cterasdk.core.query.QueryParamBuilder* method), 118
- countLimit() (*cterasdk.edge.query.QueryParamBuilder* method), 159
- CreateElement() (in module *cterasdk.convert.format*), 88
- CreateFilesWriteData (*cterasdk.edge.enum.AuditEvents* attribute), 143
- CreateFolderRC (class in *cterasdk.edge.backup*), 138
- CreateFoldersAppendData (*cterasdk.edge.enum.AuditEvents* attribute), 143
- CreateShareParam (class in

*cterasdk.core.files.common*), 93  
CriteriaBuilder (*class in cterasdk.common.types*), 85  
CRITICAL (*cterasdk.core.enum.Severity attribute*), 114  
CRITICAL (*cterasdk.edge.enum.Severity attribute*), 148  
CTERAClient (*class in cterasdk.client.cteraclient*), 79  
CTERAClientException, 182  
CTERAConnectionError, 182  
CTERAException, 182  
CTERAHost (*class in cterasdk.client.host*), 80  
CTERAPath (*class in cterasdk.core.files.path*), 94  
CTERAPath (*class in cterasdk.edge.files.path*), 135  
cterasdk  
  module, 79  
cterasdk.client  
  module, 79  
cterasdk.client.cteraclient  
  module, 79  
cterasdk.client.host  
  module, 80  
cterasdk.client.http  
  module, 81  
cterasdk.client.ssl  
  module, 83  
cterasdk.common  
  module, 83  
cterasdk.common.datetime\_utils  
  module, 83  
cterasdk.common.item  
  module, 83  
cterasdk.common.object  
  module, 83  
cterasdk.common.types  
  module, 84  
cterasdk.config  
  module, 182  
cterasdk.convert  
  module, 88  
cterasdk.convert.exception  
  module, 88  
cterasdk.convert.format  
  module, 88  
cterasdk.convert.parse  
  module, 88  
cterasdk.convert.xml\_types  
  module, 89  
cterasdk.core  
  module, 89  
cterasdk.core.activation  
  module, 95  
cterasdk.core.antivirus  
  module, 95  
cterasdk.core.base\_command  
  module, 96  
cterasdk.core.buckets  
  module, 96  
cterasdk.core.cloudfs  
  module, 97  
cterasdk.core.connection  
  module, 99  
cterasdk.core.decorator  
  module, 99  
cterasdk.core.devices  
  module, 99  
cterasdk.core.directoryservice  
  module, 101  
cterasdk.core.enum  
  module, 103  
cterasdk.core.files  
  module, 89  
cterasdk.core.files.browser  
  module, 89  
cterasdk.core.files.collaboration  
  module, 92  
cterasdk.core.files.common  
  module, 93  
cterasdk.core.files.cp  
  module, 93  
cterasdk.core.files.directory  
  module, 93  
cterasdk.core.files.file\_access  
  module, 93  
cterasdk.core.files.ln  
  module, 94  
cterasdk.core.files.ls  
  module, 94  
cterasdk.core.files.mv  
  module, 94  
cterasdk.core.files.path  
  module, 94  
cterasdk.core.files.recover  
  module, 94  
cterasdk.core.files.rename  
  module, 94  
cterasdk.core.files.rm  
  module, 95  
cterasdk.core.login  
  module, 115  
cterasdk.core.logs  
  module, 115  
cterasdk.core.plans  
  module, 120  
cterasdk.core.portals  
  module, 116  
cterasdk.core.query  
  module, 118  
cterasdk.core.remote  
  module, 121  
cterasdk.core.reports

- module, 119
- cterasdk.core.servers
  - module, 121
- cterasdk.core.session
  - module, 122
- cterasdk.core.setup
  - module, 122
- cterasdk.core.startup
  - module, 123
- cterasdk.core.syslog
  - module, 123
- cterasdk.core.taskmgr
  - module, 124
- cterasdk.core.templates
  - module, 125
- cterasdk.core.types
  - module, 127
- cterasdk.core.users
  - module, 131
- cterasdk.core.zones
  - module, 132
- cterasdk.edge
  - module, 133
  - cterasdk.edge.afp
    - module, 136
  - cterasdk.edge.aio
    - module, 136
  - cterasdk.edge.array
    - module, 136
  - cterasdk.edge.audit
    - module, 137
  - cterasdk.edge.backup
    - module, 137
  - cterasdk.edge.base\_command
    - module, 139
  - cterasdk.edge.cache
    - module, 139
  - cterasdk.edge.cli
    - module, 140
  - cterasdk.edge.config
    - module, 140
  - cterasdk.edge.connection
    - module, 141
  - cterasdk.edge.decorator
    - module, 141
  - cterasdk.edge.directoryservice
    - module, 141
  - cterasdk.edge.drive
    - module, 142
  - cterasdk.edge.enum
    - module, 143
  - cterasdk.edge.files
    - module, 133
    - cterasdk.edge.files.browser
      - module, 133
    - cterasdk.edge.files.copy
      - module, 135
    - cterasdk.edge.files.file\_access
      - module, 135
    - cterasdk.edge.files.mkdir
      - module, 135
    - cterasdk.edge.files.move
      - module, 135
    - cterasdk.edge.files.path
      - module, 135
    - cterasdk.edge.files.rm
      - module, 135
    - cterasdk.edge.firmware
      - module, 153
    - cterasdk.edge.ftp
      - module, 152
    - cterasdk.edge.groups
      - module, 153
    - cterasdk.edge.licenses
      - module, 154
    - cterasdk.edge.login
      - module, 154
    - cterasdk.edge.logs
      - module, 154
    - cterasdk.edge.mail
      - module, 155
    - cterasdk.edge.network
      - module, 155
    - cterasdk.edge.nfs
      - module, 157
    - cterasdk.edge.ntp
      - module, 157
    - cterasdk.edge.power
      - module, 159
    - cterasdk.edge.query
      - module, 159
    - cterasdk.edge.remote
      - module, 160
    - cterasdk.edge.rsync
      - module, 160
    - cterasdk.edge.services
      - module, 160
    - cterasdk.edge.session
      - module, 161
    - cterasdk.edge.shares
      - module, 162
    - cterasdk.edge.shell
      - module, 165
    - cterasdk.edge.smb
      - module, 166
    - cterasdk.edge.ssh
      - module, 158
    - cterasdk.edge.ssl
      - module, 166



module, 158  
 cteraskd.edge.support  
   module, 167  
 cteraskd.edge.sync  
   module, 168  
 cteraskd.edge.syslog  
   module, 169  
 cteraskd.edge.taskmgr  
   module, 170  
 cteraskd.edge.telnet  
   module, 170  
 cteraskd.edge.timezone  
   module, 170  
 cteraskd.edge.types  
   module, 171  
 cteraskd.edge.uri  
   module, 172  
 cteraskd.edge.users  
   module, 173  
 cteraskd.edge.volumes  
   module, 174  
 cteraskd.exception  
   module, 182  
 cteraskd.lib  
   module, 174  
 cteraskd.lib.cmd  
   module, 174  
 cteraskd.lib.consent  
   module, 175  
 cteraskd.lib.file\_access\_base  
   module, 175  
 cteraskd.lib.filesystem  
   module, 175  
 cteraskd.lib.iterator  
   module, 176  
 cteraskd.lib.platform  
   module, 176  
 cteraskd.lib.registry  
   module, 176  
 cteraskd.lib.session\_base  
   module, 176  
 cteraskd.lib.tempfile  
   module, 177  
 cteraskd.lib.tracker  
   module, 177  
 cteraskd.lib.version  
   module, 177  
 cteraskd.object  
   module, 178  
 cteraskd.object.Agent  
   module, 178  
 cteraskd.object.Gateway  
   module, 178  
 cteraskd.object.Portal

module, 180  
 cteraskd.transcript  
   module, 182  
 cteraskd.transcript.transcribe  
   module, 182  
 cttp (*cteraskd.edge.support.DebugLevel* attribute), 167  
 cttp\_data (*cteraskd.edge.support.DebugLevel* attribute), 167

## D

Daily (*cteraskd.core.enum.PlanRetention* attribute), 110  
 DateTime (*cteraskd.core.query.FilterType* attribute), 118  
 DateTimeCriteriaBuilder (class in *cteraskd.common.types*), 85  
 DateTimeUtils (class in *cteraskd.common.datetime\_utils*), 83  
 days() (*cteraskd.common.types.TimeRange* method), 87  
 DB (*cteraskd.convert.xml\_types.XMLTypes* attribute), 89  
 db (*cteraskd.edge.support.DebugLevel* attribute), 167  
 db() (*cteraskd.client.cteraclient.CTERAClient* method), 79  
 db() (*cteraskd.client.host.CTERAHost* method), 80  
 DEBUG (*cteraskd.core.enum.Severity* attribute), 114  
 DEBUG (*cteraskd.edge.enum.Severity* attribute), 149  
 debug (*cteraskd.edge.support.DebugLevel* attribute), 167  
 DebugLevel (class in *cteraskd.edge.support*), 167  
 DeduplicationMethodType (class in *cteraskd.core.enum*), 104  
 default (*cteraskd.core.antivirus.Antivirus* attribute), 95  
 default (*cteraskd.core.buckets.Buckets* attribute), 96  
 default (*cteraskd.core.cloudfs.CloudFS* attribute), 97  
 default (*cteraskd.core.devices.Devices* attribute), 100  
 default (*cteraskd.core.plans.Plans* attribute), 120  
 default (*cteraskd.core.portals.Portals* attribute), 117  
 default (*cteraskd.core.servers.Servers* attribute), 121  
 default (*cteraskd.core.templates.Templates* attribute), 126  
 default (*cteraskd.core.users.Users* attribute), 131  
 default\_class() (*cteraskd.client.host.CTERAHost* method), 80  
 default\_include (*cteraskd.edge.logs.Logs* attribute), 154  
 default\_settings() (*cteraskd.core.setup.Setup* static method), 122  
 defaultAuditEvents (*cteraskd.edge.audit.Audit* attribute), 137  
 Delete (*cteraskd.edge.enum.AuditEvents* attribute), 143  
 delete() (*cteraskd.client.cteraclient.CTERAClient* method), 79  
 delete() (*cteraskd.client.host.CTERAHost* method), 80  
 delete() (*cteraskd.client.http.HTTPClient* method), 81  
 delete() (*cteraskd.core.antivirus.AntivirusServers* method), 96  
 delete() (*cteraskd.core.buckets.Buckets* method), 96

- delete() (*cterasdk.core.cloudfs.CloudFS method*), 97
- delete() (*cterasdk.core.files.browser.CloudDrive method*), 90
- delete() (*cterasdk.core.plans.Plans method*), 120
- delete() (*cterasdk.core.portals.Portals method*), 117
- delete() (*cterasdk.core.templates.Templates method*), 126
- delete() (*cterasdk.core.users.Users method*), 131
- delete() (*cterasdk.core.zones.Zones method*), 133
- delete() (*cterasdk.edge.array.Array method*), 136
- delete() (*cterasdk.edge.files.browser.FileBrowser method*), 133
- delete() (*cterasdk.edge.shares.Shares method*), 163
- delete() (*cterasdk.edge.users.Users method*), 173
- delete() (*cterasdk.edge.volumes.Volumes method*), 174
- delete() (*in module cterasdk.core.files.rm*), 95
- delete() (*in module cterasdk.edge.files.rm*), 135
- delete\_all() (*cterasdk.edge.array.Array method*), 136
- delete\_all() (*cterasdk.edge.volumes.Volumes method*), 174
- delete\_attr() (*in module cterasdk.common.object*), 83
- delete\_attrs() (*in module cterasdk.common.object*), 83
- delete\_multi() (*cterasdk.core.files.browser.CloudDrive method*), 90
- delete\_multi() (*in module cterasdk.core.files.rm*), 95
- Deleted (*cterasdk.core.enum.ListFilter attribute*), 108
- Deleted (*cterasdk.core.enum.PlanRetention attribute*), 110
- DeleteSubfoldersAndFiles (*cterasdk.edge.enum.AuditEvents attribute*), 143
- desktops() (*cterasdk.core.devices.Devices method*), 100
- Device (*class in cterasdk.common.object*), 83
- Device (*cterasdk.core.enum.OriginType attribute*), 109
- device() (*cterasdk.core.devices.Devices method*), 100
- device() (*cterasdk.core.logs.Logs method*), 115
- device\_config() (*cterasdk.core.files.browser.Backups method*), 89
- Devices (*class in cterasdk.core.devices*), 99
- devices() (*cterasdk.core.devices.Devices method*), 100
- DeviceType (*class in cterasdk.core.enum*), 104
- df() (*cterasdk.config.Logging static method*), 182
- DG (*cterasdk.core.enum.CollaboratorType attribute*), 104
- DG (*cterasdk.edge.enum.PrincipalType attribute*), 147
- diagnose() (*cterasdk.edge.network.Network method*), 155
- DirectoryEntryFactory (*class in cterasdk.common.types*), 85
- DirectorySearchEntityType (*class in cterasdk.core.enum*), 105
- DirectoryService (*class in cterasdk.core.directoryservice*), 101
- DirectoryService (*class in cterasdk.edge.directoryservice*), 141
- DirectoryServiceFetchMode (*class in cterasdk.core.enum*), 105
- DirectoryServiceType (*class in cterasdk.core.enum*), 105
- DirEntry (*class in cterasdk.common.types*), 85
- disable() (*cterasdk.config.Logging static method*), 182
- disable() (*cterasdk.core.syslog.Syslog method*), 123
- disable() (*cterasdk.edge.afp.AFP method*), 136
- disable() (*cterasdk.edge.aio.AIO method*), 136
- disable() (*cterasdk.edge.audit.Audit method*), 137
- disable() (*cterasdk.edge.cache.Cache method*), 139
- disable() (*cterasdk.edge.ftp.FTP method*), 152
- disable() (*cterasdk.edge.mail.Mail method*), 155
- disable() (*cterasdk.edge.nfs.NFS method*), 157
- disable() (*cterasdk.edge.ntp.NTP method*), 157
- disable() (*cterasdk.edge.rsync.RSync method*), 160
- disable() (*cterasdk.edge.smb.SMB method*), 166
- disable() (*cterasdk.edge.ssh.SSH method*), 158
- disable() (*cterasdk.edge.syslog.Syslog method*), 169
- disable() (*cterasdk.edge.telnet.Telnet method*), 170
- disable\_abe() (*cterasdk.edge.smb.SMB method*), 166
- disable\_http() (*cterasdk.edge.ssl.SSL method*), 158
- disable\_remote\_access() (*cterasdk.edge.session.Session method*), 161
- disable\_sso() (*cterasdk.edge.services.Services method*), 161
- disable\_wizard() (*cterasdk.edge.config.Config method*), 140
- Disabled (*cterasdk.core.enum.Mode attribute*), 108
- Disabled (*cterasdk.core.enum.Role attribute*), 112
- Disabled (*cterasdk.edge.enum.CIFSPacketSigning attribute*), 144
- Disabled (*cterasdk.edge.enum.ClientSideCaching attribute*), 145
- Disabled (*cterasdk.edge.enum.Mode attribute*), 146
- Disabled (*cterasdk.edge.enum.OperationMode attribute*), 146
- disconnect() (*cterasdk.core.directoryservice.DirectoryService method*), 101
- disconnect() (*cterasdk.edge.directoryservice.DirectoryService method*), 142
- disconnect() (*cterasdk.edge.services.Services method*), 161
- Disconnected (*cterasdk.edge.enum.ServicesConnectionState attribute*), 148
- DisconnectedPortal (*cterasdk.edge.enum.SyncStatus attribute*), 150
- dispatch() (*cterasdk.client.http.HttpClientBase method*), 82
- DLP (*cterasdk.core.enum.ICAPServices attribute*), 107
- dns (*cterasdk.edge.support.DebugLevel attribute*), 167

- Documents (*cterasdk.edge.enum.ClientSideCaching attribute*), 145
- domain\_group() (*cterasdk.core.types.ShareRecipient static method*), 129
- domain\_user() (*cterasdk.core.types.ShareRecipient static method*), 129
- DomainControllers (*class in cterasdk.core.types*), 127
- domains() (*cterasdk.core.directoryservice.DirectoryService method*), 102
- domains() (*cterasdk.edge.directoryservice.DirectoryService method*), 142
- Download (*cterasdk.edge.enum.Traffic attribute*), 151
- download() (*cterasdk.client.cteraclient.CTERAClient method*), 79
- download() (*cterasdk.common.types.ThrottlingRuleBuilder method*), 87
- download() (*cterasdk.core.files.browser.FileBrowser method*), 91
- download() (*cterasdk.edge.files.browser.FileBrowser method*), 133
- download() (*cterasdk.lib.file\_access\_base.FileAccessBase method*), 175
- download\_as\_zip() (*cterasdk.core.files.browser.FileBrowser method*), 92
- download\_as\_zip() (*cterasdk.edge.files.browser.FileBrowser method*), 134
- download\_as\_zip() (*cterasdk.lib.file\_access\_base.FileAccessBase method*), 175
- download\_zip() (*cterasdk.client.cteraclient.CTERAClient method*), 79
- download\_zip() (*cterasdk.client.host.CTERAHost method*), 80
- Drive (*class in cterasdk.edge.drive*), 142
- DU (*cterasdk.core.enum.CollaboratorType attribute*), 104
- DU (*cterasdk.edge.enum.PrincipalType attribute*), 147
- ## E
- Eager (*cterasdk.core.enum.DirectoryServiceFetchMode attribute*), 105
- Edge\_6 (*cterasdk.core.enum.Platform attribute*), 111
- Edge\_7 (*cterasdk.core.enum.Platform attribute*), 111
- Email (*cterasdk.core.enum.ProtectionLevel attribute*), 112
- EMERGENCY (*cterasdk.core.enum.Severity attribute*), 114
- EMERGENCY (*cterasdk.edge.enum.Severity attribute*), 149
- enable() (*cterasdk.config.Logging static method*), 182
- enable() (*cterasdk.core.syslog.Syslog method*), 123
- enable() (*cterasdk.edge.aio.AIO method*), 136
- enable() (*cterasdk.edge.audit.Audit method*), 137
- enable() (*cterasdk.edge.cache.Cache method*), 139
- enable() (*cterasdk.edge.ftp.FTP method*), 152
- enable() (*cterasdk.edge.mail.Mail method*), 155
- enable() (*cterasdk.edge.nfs.NFS method*), 157
- enable() (*cterasdk.edge.ntp.NTP method*), 157
- enable() (*cterasdk.edge.rsync.RSync method*), 160
- enable() (*cterasdk.edge.smb.SMB method*), 166
- enable() (*cterasdk.edge.ssh.SSH method*), 158
- enable() (*cterasdk.edge.syslog.Syslog method*), 169
- enable() (*cterasdk.edge.telnet.Telnet method*), 170
- enable\_abe() (*cterasdk.edge.smb.SMB method*), 166
- enable\_dhcp() (*cterasdk.edge.network.Network method*), 155
- enable\_http() (*cterasdk.edge.ssl.SSL method*), 158
- enable\_remote\_access() (*cterasdk.edge.session.Session method*), 161
- enable\_sso() (*cterasdk.edge.services.Services method*), 161
- enable\_wizard() (*cterasdk.edge.config.Config method*), 140
- Enabled (*cterasdk.core.enum.Mode attribute*), 108
- Enabled (*cterasdk.edge.enum.Mode attribute*), 146
- encoded\_fullpath() (*cterasdk.core.files.path.CTERAPath method*), 94
- encoded\_fullpath() (*cterasdk.edge.files.path.CTERAPath method*), 135
- encoded\_parent() (*cterasdk.core.files.path.CTERAPath method*), 94
- encoded\_parent() (*cterasdk.edge.files.path.CTERAPath method*), 135
- EncryptionMode (*class in cterasdk.edge.backup*), 138
- end() (*cterasdk.common.types.TimeRange method*), 87
- endswith() (*cterasdk.common.types.StringCriteriaBuilder method*), 86
- EndsWithOperator (*class in cterasdk.common.types*), 85
- EndUser (*cterasdk.core.enum.Role attribute*), 112
- EnvironmentVariables (*class in cterasdk.core.enum*), 106
- eq() (*cterasdk.core.query.FilterBuilder method*), 118
- EQUALS (*cterasdk.core.query.Restriction attribute*), 119
- equals() (*cterasdk.common.types.StringCriteriaBuilder method*), 86
- ERROR (*cterasdk.core.enum.Severity attribute*), 114
- ERROR (*cterasdk.edge.enum.Severity attribute*), 149
- error (*cterasdk.edge.support.DebugLevel attribute*), 167
- error\_abort (*cterasdk.edge.support.DebugLevel attribute*), 167
- ErrorStatus, 177
- ESET (*cterasdk.core.enum.AntivirusType attribute*), 103
- EV128 (*cterasdk.core.enum.PlanItem attribute*), 110
- EV128 (*cterasdk.edge.enum.License attribute*), 146
- EV16 (*cterasdk.core.enum.PlanItem attribute*), 110
- EV16 (*cterasdk.edge.enum.License attribute*), 146
- EV32 (*cterasdk.core.enum.PlanItem attribute*), 110
- EV32 (*cterasdk.edge.enum.License attribute*), 146
- EV4 (*cterasdk.core.enum.PlanItem attribute*), 110
- EV4 (*cterasdk.edge.enum.License attribute*), 146

- EV64 (*cterasdk.core.enum.PlanItem* attribute), 110
- EV64 (*cterasdk.edge.enum.License* attribute), 146
- EV8 (*cterasdk.core.enum.PlanItem* attribute), 110
- EV8 (*cterasdk.edge.enum.License* attribute), 146
- Everyone (*cterasdk.edge.enum.LocalGroup* attribute), 146
- evictor (*cterasdk.edge.support.DebugLevel* attribute), 167
- evictor\_verbose (*cterasdk.edge.support.DebugLevel* attribute), 167
- exclude\_files() (*cterasdk.edge.sync.Sync* method), 168
- execute() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- execute() (*cterasdk.client.host.CTERAHost* method), 80
- ExhaustedException, 183
- exists() (*cterasdk.lib.filesystem.FileSystem* static method), 175
- expanduser() (*cterasdk.lib.filesystem.FileSystem* static method), 175
- expire\_in() (*cterasdk.core.types.ShareRecipient* method), 129
- expire\_on() (*cterasdk.core.types.ShareRecipient* method), 129
- export() (*cterasdk.edge.config.Config* method), 140
- EXT (*cterasdk.core.enum.CollaboratorType* attribute), 104
- extensions() (*cterasdk.common.types.FileFilterBuilder* static method), 85
- external() (*cterasdk.core.types.ShareRecipient* static method), 129
- ## F
- FAIL (*cterasdk.edge.firmware.UploadTaskStatus* attribute), 153
- Failed (*cterasdk.core.enum.SetupWizardStatus* attribute), 113
- Failed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- Failed (*cterasdk.edge.enum.TaskStatus* attribute), 150
- failed() (*cterasdk.lib.tracker.StatusTracker* method), 177
- FailedFilesInReadOnlyFolder (*cterasdk.edge.enum.SyncStatus* attribute), 150
- fetch() (*cterasdk.core.directoryservice.DirectoryService* method), 102
- fetch\_resources() (in module *cterasdk.core.files.ls*), 94
- file\_descriptor() (*cterasdk.client.cteraclient.CTERAClient* static method), 79
- FileAccess (class in *cterasdk.core.files.file\_access*), 93
- FileAccess (class in *cterasdk.edge.files.file\_access*), 135
- FileAccessBase (class in *cterasdk.lib.file\_access\_base*), 175
- FileAccessMode (class in *cterasdk.core.enum*), 107
- FileAccessMode (class in *cterasdk.edge.enum*), 145
- FileBrowser (class in *cterasdk.core.files.browser*), 91
- FileBrowser (class in *cterasdk.edge.files.browser*), 133
- FileEntry (class in *cterasdk.common.types*), 85
- FileFilterBuilder (class in *cterasdk.common.types*), 85
- filers() (*cterasdk.core.devices.Devices* method), 100
- files (*cterasdk.edge.support.DebugLevel* attribute), 167
- files() (in module *cterasdk.edge.uri*), 172
- FileSystem (class in *cterasdk.lib.filesystem*), 175
- FileSystemException, 183
- Filter (class in *cterasdk.core.query*), 118
- FilterBackupSet (class in *cterasdk.common.types*), 86
- FilterBuilder (class in *cterasdk.core.query*), 118
- FilterType (class in *cterasdk.core.query*), 118
- find() (*cterasdk.core.cloudfs.CloudFS* method), 97
- find\_attr() (in module *cterasdk.common.object*), 84
- Finish (*cterasdk.core.enum.SetupWizardStage* attribute), 113
- Firmware (class in *cterasdk.edge.firmware*), 153
- FIRMWARE (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89
- First (*cterasdk.core.enum.PlanCriteria* attribute), 109
- first\_name() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128
- FixedBlockSize (*cterasdk.core.enum.DeduplicationMethodType* attribute), 104
- fmt() (*cterasdk.config.Logging* static method), 182
- folder\_groups() (*cterasdk.core.reports.Reports* method), 119
- FolderAlreadyExists (*cterasdk.edge.backup.CreateFolderRC* attribute), 138
- folders() (*cterasdk.core.reports.Reports* method), 119
- Forbidden, 135
- force\_eviction() (*cterasdk.edge.cache.Cache* method), 139
- form\_data() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- form\_data() (*cterasdk.client.host.CTERAHost* method), 80
- format() (*cterasdk.edge.drive.Drive* method), 142
- format\_all() (*cterasdk.edge.drive.Drive* method), 142
- Formatting (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- from\_collaborator() (*cterasdk.core.types.PortalAccount* static method), 129
- from\_server\_object()



- (*cterasdk.common.types.ThrottlingRule* static method), 87
- `from_server_object()`  
(*cterasdk.edge.types.NFSv3AccessControlEntry* static method), 171
- `from_server_object()`  
(*cterasdk.edge.types.ShareAccessControlEntry* static method), 171
- `from_server_object()`  
(*cterasdk.edge.types.UserGroupEntry* static method), 172
- `fromjsonstr()` (in module *cterasdk.convert.parse*), 88
- `fromValue()` (*cterasdk.core.query.FilterType* static method), 118
- `fromxmlstr()` (*cterasdk.client.cteraclient.CTERAClient* static method), 79
- `fromxmlstr()` (in module *cterasdk.convert.parse*), 88
- FTP (class in *cterasdk.edge.ftp*), 152
- `fullpath()` (*cterasdk.core.files.path.CTERAPath* method), 94
- `fullpath()` (*cterasdk.edge.files.path.CTERAPath* method), 135
- ## G
- Gateway (class in *cterasdk.object.Gateway*), 178
- Gateways (*cterasdk.core.enum.DeviceType* attribute), 105
- `ge()` (*cterasdk.core.query.FilterBuilder* method), 118
- `generate_code()` (*cterasdk.core.activation.Activation* method), 95
- GenericS3 (class in *cterasdk.core.types*), 127
- `get()` (*cterasdk.client.cteraclient.CTERAClient* method), 79
- `get()` (*cterasdk.client.host.CTERAHost* method), 80
- `get()` (*cterasdk.client.http.HTTPClient* method), 81
- `get()` (*cterasdk.config.Logging* static method), 182
- `get()` (*cterasdk.core.antivirus.AntivirusServers* method), 96
- `get()` (*cterasdk.core.buckets.Buckets* method), 97
- `get()` (*cterasdk.core.cloudfs.CloudFS* method), 98
- `get()` (*cterasdk.core.logs.Logs* method), 116
- `get()` (*cterasdk.core.plans.Plans* method), 121
- `get()` (*cterasdk.core.portals.Portals* method), 117
- `get()` (*cterasdk.core.servers.Servers* method), 121
- `get()` (*cterasdk.core.templates.Templates* method), 126
- `get()` (*cterasdk.core.users.Users* method), 131
- `get()` (*cterasdk.core.zones.Zones* method), 133
- `get()` (*cterasdk.edge.array.Array* method), 137
- `get()` (*cterasdk.edge.drive.Drive* method), 142
- `get()` (*cterasdk.edge.groups.Groups* method), 153
- `get()` (*cterasdk.edge.licenses.Licenses* method), 154
- `get()` (*cterasdk.edge.licenses.LocalLicenses* method), 154
- `get()` (*cterasdk.edge.shares.Shares* method), 163
- `get()` (*cterasdk.edge.users.Users* method), 173
- `get()` (*cterasdk.edge.volumes.Volumes* method), 174
- `get()` (*cterasdk.lib.registry.Registry* method), 176
- `get_access_control()`  
(*cterasdk.core.directoryservice.DirectoryService* method), 102
- `get_access_type()` (*cterasdk.edge.shares.Shares* method), 163
- `get_acl()` (*cterasdk.edge.shares.Shares* method), 163
- `get_advanced_mapping()`  
(*cterasdk.core.directoryservice.DirectoryService* method), 102
- `get_attr()` (in module *cterasdk.common.object*), 84
- `get_configuration()` (*cterasdk.core.syslog.Syslog* method), 124
- `get_configuration()` (*cterasdk.edge.ftp.FTP* method), 152
- `get_configuration()` (*cterasdk.edge.nfs.NFS* method), 157
- `get_configuration()` (*cterasdk.edge.ntp.NTP* method), 157
- `get_configuration()` (*cterasdk.edge.rsync.RSync* method), 160
- `get_configuration()` (*cterasdk.edge.smb.SMB* method), 166
- `get_configuration()` (*cterasdk.edge.syslog.Syslog* method), 169
- `get_connected_domain()`  
(*cterasdk.core.directoryservice.DirectoryService* method), 102
- `get_connected_domain()`  
(*cterasdk.edge.directoryservice.DirectoryService* method), 142
- `get_default_role()` (*cterasdk.core.directoryservice.DirectoryService* method), 25, 102
- `get_dirpath()` (*cterasdk.lib.filesystem.FileSystem* method), 175
- `get_expiration_date()`  
(*cterasdk.common.datetime\_utils.DateTimeUtils* static method), 83
- `get_hostname()` (*cterasdk.edge.config.Config* method), 140
- `get_linux_avoid_using_fanotify()`  
(*cterasdk.edge.sync.Sync* method), 168
- `get_local_file_info()`  
(*cterasdk.lib.filesystem.FileSystem* static method), 175
- `get_location()` (*cterasdk.edge.config.Config* method), 140
- `get_multi()` (*cterasdk.client.cteraclient.CTERAClient* method), 79
- `get_multi()` (*cterasdk.client.host.CTERAHost* method), 80
- `get_policy()` (*cterasdk.core.plans.PlanAutoAssignPolicy*

- method), 120
- get\_policy() (cterasdk.core.templates.TemplateAutoAssignPolicy attribute), 119
- method), 125
- get\_policy() (cterasdk.edge.sync.CloudSyncBandwidthThrottling attribute), 119
- method), 168
- get\_replication\_candidates() (cterasdk.core.setup.Setup method), 122
- get\_resource\_info() (in module cterasdk.core.files.common), 93
- get\_screened\_file\_types() (cterasdk.edge.shares.Shares method), 163
- get\_session\_id() (cterasdk.client.cteraclient.CTERAClient method), 79
- get\_session\_id() (cterasdk.client.host.CTERAHost method), 80
- get\_session\_id() (cterasdk.client.http.HttpClientBase method), 82
- get\_setup\_status() (cterasdk.core.setup.Setup method), 122
- get\_share\_info() (cterasdk.core.files.browser.CloudDrive method), 90
- get\_share\_info() (in module cterasdk.core.files.collaboration), 92
- get\_static\_domain\_controller() (cterasdk.edge.directoryservice.DirectoryService method), 142
- get\_status() (cterasdk.edge.drive.Drive method), 142
- get\_status() (cterasdk.edge.network.Network method), 156
- get\_status() (cterasdk.edge.services.Services method), 161
- get\_status() (cterasdk.edge.sync.Sync method), 168
- get\_storage\_ca() (cterasdk.edge.ssl.SSL method), 158
- get\_support\_report() (cterasdk.edge.support.Support method), 168
- get\_task\_status() (cterasdk.core.taskmgr.Task method), 124
- get\_task\_status() (cterasdk.edge.taskmgr.Task method), 170
- get\_timezone() (cterasdk.edge.timezone.Timezone method), 170
- get\_trusted\_nfs\_clients() (cterasdk.edge.shares.Shares method), 163
- getcode() (cterasdk.client.http.HTTPResponse method), 82
- GetFoldersList (cterasdk.edge.enum.BackupConfStatusID attribute), 144
- geturi() (in module cterasdk.client.http), 83
- geturl() (cterasdk.client.http.HTTPResponse method), 82
- GlobalAdmin (class in cterasdk.object.Portal), 180
- Google (class in cterasdk.core.types), 128
- Google (cterasdk.core.enum.BucketType attribute), 103
- GREATER\_EQUALS (cterasdk.core.query.Restriction attribute), 119
- GREATER\_THAN (cterasdk.core.query.Restriction attribute), 119
- Group (cterasdk.core.enum.DirectorySearchEntityType attribute), 105
- Group (cterasdk.core.enum.PortalAccountType attribute), 112
- GroupAccount (class in cterasdk.core.types), 128
- Groups (class in cterasdk.edge.groups), 153
- Groups (cterasdk.core.enum.PlanCriteria attribute), 109
- Groups (cterasdk.core.enum.SearchType attribute), 113
- Groups (cterasdk.core.enum.TemplateCriteria attribute), 115
- groups() (cterasdk.core.types.TemplateCriteriaBuilder static method), 130
- gt() (cterasdk.core.query.FilterBuilder method), 118
- ## H
- host (cterasdk.edge.types.TCPCConnectResult property), 172
- host (cterasdk.edge.types.TCPService property), 172
- host() (cterasdk.client.host.NetworkHost method), 81
- Hostname (cterasdk.core.enum.TemplateCriteria attribute), 115
- hostname() (cterasdk.core.types.TemplateCriteriaBuilder static method), 130
- HostUnreachable, 183
- Hourly (cterasdk.core.enum.PlanRetention attribute), 110
- http (cterasdk.edge.support.DebugLevel attribute), 167
- HTTPBucket (class in cterasdk.core.types), 128
- HTTPClient (class in cterasdk.client.http), 81
- HttpClientBase (class in cterasdk.client.http), 82
- HttpRequest (class in cterasdk.client.http), 82
- HttpRequestCopy (class in cterasdk.client.http), 82
- HttpRequestCopyMove (class in cterasdk.client.http), 82
- HttpRequestDelete (class in cterasdk.client.http), 82
- HttpRequestGet (class in cterasdk.client.http), 82
- HttpRequestMkcol (class in cterasdk.client.http), 82
- HttpRequestMove (class in cterasdk.client.http), 82
- HttpRequestPost (class in cterasdk.client.http), 82
- HttpRequestPut (class in cterasdk.client.http), 82
- HTTPException, 82
- HTTPResponse (class in cterasdk.client.http), 82
- https() (cterasdk.client.host.NetworkHost method), 81

- I**
- ICAPServices (class in *cterasdk.core.enum*), 107
  - ICOS (class in *cterasdk.core.types*), 128
  - ICOS (*cterasdk.core.enum.BucketType* attribute), 103
  - ID (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89
  - IfClientAgrees (*cterasdk.edge.enum.CIFSPacketSigning* attribute), 144
  - ifconfig() (*cterasdk.edge.network.Network* method), 156
  - import\_certificate() (*cterasdk.edge.ssl.SSL* method), 158
  - import\_config() (*cterasdk.edge.config.Config* method), 140
  - import\_storage\_ca() (*cterasdk.edge.ssl.SSL* method), 158
  - IN\_PROGRESS (*cterasdk.edge.firmware.UploadTaskStatus* attribute), 153
  - in\_tenant\_context() (*cterasdk.core.session.Session* method), 122
  - Inactive (*cterasdk.lib.session\_base.SessionStatus* attribute), 176
  - include() (*cterasdk.common.types.ListCriteriaBuilder* method), 86
  - include() (*cterasdk.core.query.QueryParamBuilder* method), 118
  - include() (*cterasdk.edge.query.QueryParamBuilder* method), 159
  - include\_classname() (*cterasdk.core.query.QueryParamBuilder* method), 118
  - include\_classname() (*cterasdk.core.query.QueryParams* method), 119
  - include\_classname() (*cterasdk.edge.query.QueryParam* method), 159
  - IncorrectPassphrase, 138
  - increment() (*cterasdk.core.query.QueryParams* method), 119
  - increment() (*cterasdk.edge.query.QueryParam* method), 159
  - increment() (*cterasdk.lib.tracker.StatusTracker* method), 177
  - index (*cterasdk.edge.support.DebugLevel* attribute), 167
  - infer() (*cterasdk.edge.licenses.Licenses* static method), 154
  - INFO (*cterasdk.core.enum.Severity* attribute), 114
  - INFO (*cterasdk.edge.enum.Severity* attribute), 149
  - info (*cterasdk.edge.support.DebugLevel* attribute), 167
  - info() (*cterasdk.edge.login.Login* method), 154
  - init\_application\_server() (*cterasdk.core.setup.Setup* method), 122
  - init\_master() (*cterasdk.core.setup.Setup* method), 123
  - init\_replication\_server() (*cterasdk.core.setup.Setup* method), 123
  - initialized (*cterasdk.object.Gateway.Gateway* property), 179
  - Initializing (*cterasdk.lib.session\_base.SessionStatus* attribute), 176
  - initializing() (*cterasdk.lib.session\_base.SessionBase* method), 176
  - InitializingConnection (*cterasdk.edge.enum.SyncStatus* attribute), 150
  - InputError, 183
  - instance() (*cterasdk.core.files.common.ActionResourcesParam* static method), 93
  - instance() (*cterasdk.core.files.common.CreateShareParam* static method), 93
  - instance() (*cterasdk.core.files.common.SrcDstParam* static method), 93
  - instance() (*cterasdk.lib.filesystem.FileSystem* static method), 175
  - instance() (*cterasdk.lib.platform.Platform* static method), 176
  - instance() (*cterasdk.lib.registry.Registry* static method), 176
  - instance() (*cterasdk.lib.version.Version* static method), 177
  - Integer (*cterasdk.core.query.FilterType* attribute), 118
  - IntegerCriteriaBuilder (class in *cterasdk.common.types*), 86
  - InternalError (*cterasdk.edge.enum.SyncStatus* attribute), 150
  - InternalServerError (*cterasdk.edge.backup.AttachRC* attribute), 137
  - InternalServerError (*cterasdk.edge.backup.CreateFolderRC* attribute), 138
  - interval() (*cterasdk.common.types.BackupScheduleBuilder* static method), 84
  - IntRefFilter (*cterasdk.core.query.FilterType* attribute), 118
  - InvalidAverageBlockSize (*cterasdk.edge.enum.SyncStatus* attribute), 150
  - InvalidConfiguration (*cterasdk.edge.enum.SyncStatus* attribute), 150
  - InvalidName, 93
  - InvalidPath, 93
  - ipconfig() (*cterasdk.edge.network.Network* method), 156
  - iperf() (*cterasdk.edge.network.Network* method), 156
  - IPProtocol (class in *cterasdk.core.enum*), 107
  - IPProtocol (class in *cterasdk.edge.enum*), 145

is\_configured() (*cterasdk.edge.backup.Backup* method), 138  
 is\_disabled() (*cterasdk.edge.afp.AFP* method), 136  
 is\_disabled() (*cterasdk.edge.ftp.FTP* method), 152  
 is\_disabled() (*cterasdk.edge.nfs.NFS* method), 157  
 is\_disabled() (*cterasdk.edge.rsync.RSync* method), 160  
 is\_disabled() (*cterasdk.edge.sync.Sync* method), 168  
 is\_enabled() (*cterasdk.core.syslog.Syslog* method), 124  
 is\_enabled() (*cterasdk.edge.aio.AIO* method), 136  
 is\_enabled() (*cterasdk.edge.cache.Cache* method), 139  
 is\_enabled() (*cterasdk.edge.sync.Sync* method), 168  
 is\_global\_admin() (*cterasdk.core.session.Session* method), 122  
 is\_http\_disabled() (*cterasdk.edge.ssl.SSL* method), 158  
 is\_http\_enabled() (*cterasdk.edge.ssl.SSL* method), 158  
 is\_local (*cterasdk.core.types.PortalAccount* property), 129  
 is\_local\_auth() (*cterasdk.lib.session\_base.SessionBase* method), 176  
 is\_nosession() (in module *cterasdk.edge.decorator*), 141  
 is\_open (*cterasdk.edge.types.TCPConnectResult* property), 172  
 is\_wizard\_enabled() (*cterasdk.edge.config.Config* method), 140  
 IsEncrypted (*cterasdk.edge.backup.AttachRC* attribute), 137  
 isoneof() (*cterasdk.common.types.StringCriteriaBuilder* method), 86  
 IsOneOfOperator (class in *cterasdk.common.types*), 86  
 IsOperator (class in *cterasdk.common.types*), 86  
 Item (class in *cterasdk.common.item*), 83  
 ItemExists, 93, 135  
 Iterator (class in *cterasdk.lib.iterator*), 176  
 iterator() (*cterasdk.object.Portal.Portal* method), 181  
 iterator() (in module *cterasdk.core.query*), 119

## J

JBOD (*cterasdk.edge.enum.RAIDLevel* attribute), 147  
 join() (*cterasdk.exception.CTERAException* method), 182  
 join() (*cterasdk.lib.filesystem.FileSystem* static method), 175  
 joinpath() (*cterasdk.core.files.path.CTERAPath* method), 94  
 joinpath() (*cterasdk.edge.files.path.CTERAPath* method), 135

## K

KeyRequired (*cterasdk.edge.enum.VolumeStatus* attribute), 151

## L

Last (*cterasdk.core.enum.PlanCriteria* attribute), 109  
 last\_modified() (*cterasdk.common.types.FileFilterBuilder* static method), 85  
 last\_name() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128  
 Lazy (*cterasdk.core.enum.DirectoryServiceFetchMode* attribute), 105  
 LDAP (*cterasdk.core.enum.DirectoryServiceType* attribute), 106  
 le() (*cterasdk.core.query.FilterBuilder* method), 118  
 LESS\_EQUALS (*cterasdk.core.query.Restriction* attribute), 119  
 LESS\_THAN (*cterasdk.core.query.Restriction* attribute), 119  
 less\_than() (*cterasdk.common.types.IntegerCriteriaBuilder* method), 86  
 LessThanOperator (class in *cterasdk.common.types*), 86  
 LG (*cterasdk.core.enum.CollaboratorType* attribute), 104  
 LG (*cterasdk.edge.enum.PrincipalType* attribute), 147  
 License (class in *cterasdk.edge.enum*), 145  
 license (*cterasdk.edge.support.DebugLevel* attribute), 167  
 Licenses (class in *cterasdk.edge.licenses*), 154  
 LIKE (*cterasdk.core.query.Restriction* attribute), 119  
 like() (*cterasdk.core.query.FilterBuilder* method), 118  
 Linux (*cterasdk.core.enum.Platform* attribute), 111  
 linux() (*cterasdk.core.types.TemplateScript* static method), 130  
 LIST (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89  
 list\_buckets() (*cterasdk.core.buckets.Buckets* method), 97  
 list\_dir() (in module *cterasdk.core.files.ls*), 94  
 list\_domain\_users() (*cterasdk.core.users.Users* method), 131  
 list\_domains() (*cterasdk.core.users.Users* method), 132  
 list\_folder\_groups() (*cterasdk.core.cloudfs.CloudFS* method), 98  
 list\_folders() (*cterasdk.core.cloudfs.CloudFS* method), 98  
 list\_local\_users() (*cterasdk.core.users.Users* method), 132  
 list\_plans() (*cterasdk.core.plans.Plans* method), 121  
 list\_servers() (*cterasdk.core.antivirus.Antivirus* method), 95



- list\_servers() (cterasdk.core.servers.Servers method), 121
- list\_templates() (cterasdk.core.templates.Templates method), 126
- list\_tenants() (cterasdk.core.portals.Portals method), 117
- ListCriteriaBuilder (class in cterasdk.common.types), 86
- ListFilter (class in cterasdk.core.enum), 107
- ListFolderReadData (cterasdk.edge.enum.AuditEvents attribute), 143
- load\_config() (cterasdk.edge.config.Config method), 140
- Local (cterasdk.edge.session.SessionType attribute), 161
- local() (cterasdk.edge.session.Session method), 161
- local() (in module cterasdk.edge.uri), 172
- local\_group() (cterasdk.core.types.ShareRecipient static method), 129
- local\_user() (cterasdk.core.types.ShareRecipient static method), 129
- LocalDirectoryNotFound, 183
- LocalFileNotFound, 183
- LocalGroup (class in cterasdk.edge.enum), 146
- LocalLicenses (class in cterasdk.edge.licenses), 154
- LocalPathNotFound, 183
- LOCATION (cterasdk.convert.xml\_types.XMLTypes attribute), 89
- LocationType (class in cterasdk.core.enum), 108
- Logging (class in cterasdk.config), 182
- Login (class in cterasdk.core.login), 115
- Login (class in cterasdk.edge.login), 154
- login() (cterasdk.client.host.CTERAHost method), 80
- login() (cterasdk.core.login.Login method), 115
- login() (cterasdk.edge.login.Login method), 154
- login() (in module cterasdk.edge.remote), 160
- logout() (cterasdk.client.host.CTERAHost method), 80
- logout() (cterasdk.core.login.Login method), 115
- logout() (cterasdk.edge.login.Login method), 154
- Logs (class in cterasdk.core.logs), 115
- Logs (class in cterasdk.edge.logs), 154
- logs() (cterasdk.edge.logs.Logs method), 154
- LogTopic (class in cterasdk.core.enum), 108
- ls() (cterasdk.core.files.browser.FileBrowser method), 92
- ls() (cterasdk.edge.files.browser.FileBrowser static method), 134
- ls() (in module cterasdk.core.files.ls), 94
- lt() (cterasdk.core.query.FilterBuilder method), 118
- LU (cterasdk.core.enum.CollaboratorType attribute), 104
- LU (cterasdk.edge.enum.PrincipalType attribute), 147
- LVM (cterasdk.edge.enum.RAIDLevel attribute), 147
- M**
- mac() (cterasdk.core.types.TemplateScript static method), 130
- Mail (class in cterasdk.edge.mail), 155
- make\_local\_files\_dir() (cterasdk.object.Gateway.Gateway static method), 179
- Manual (cterasdk.edge.enum.ClientSideCaching attribute), 145
- Master (cterasdk.core.enum.ServerMode attribute), 113
- McAfeeVSES (cterasdk.core.enum.AntivirusType attribute), 103
- McAfeeWG (cterasdk.core.enum.AntivirusType attribute), 103
- Microsoft (cterasdk.core.enum.DirectoryServiceType attribute), 106
- mkcol() (cterasdk.client.cteraclient.CTERAClient method), 79
- mkcol() (cterasdk.client.host.CTERAHost method), 80
- mkcol() (cterasdk.client.http.HTTPClient method), 81
- mkdir() (cterasdk.core.cloudfs.CloudFS method), 98
- mkdir() (cterasdk.core.files.browser.CloudDrive method), 90
- mkdir() (cterasdk.edge.files.browser.FileBrowser method), 134
- mkdir() (cterasdk.lib.tempfile.TempfileServices static method), 177
- mkdir() (in module cterasdk.core.files.directory), 93
- mkdir() (in module cterasdk.edge.files.mkdir), 135
- mkfg() (cterasdk.core.cloudfs.CloudFS method), 98
- mkfile() (cterasdk.lib.tempfile.TempfileServices static method), 177
- mlink() (cterasdk.core.files.browser.FileBrowser method), 92
- mlink() (in module cterasdk.core.files.ln), 94
- mkpath() (cterasdk.core.files.browser.FileBrowser method), 92
- mkpath() (cterasdk.edge.files.browser.FileBrowser static method), 134
- Mode (class in cterasdk.core.enum), 108
- Mode (class in cterasdk.edge.enum), 146
- modify() (cterasdk.core.buckets.Buckets method), 97
- modify() (cterasdk.core.plans.Plans method), 121
- modify() (cterasdk.core.servers.Servers method), 122
- modify() (cterasdk.core.syslog.Syslog method), 124
- modify() (cterasdk.core.users.Users method), 132
- modify() (cterasdk.edge.ftp.FTP method), 152
- modify() (cterasdk.edge.nfs.NFS method), 157
- modify() (cterasdk.edge.rsync.RSync method), 160
- modify() (cterasdk.edge.shares.Shares method), 163
- modify() (cterasdk.edge.smb.SMB method), 166
- modify() (cterasdk.edge.syslog.Syslog method), 169
- modify() (cterasdk.edge.users.Users method), 173
- modify() (cterasdk.edge.volumes.Volumes method), 174
- module  
cterasdk, 79

- cterasdk.client, 79
- cterasdk.client.cteraclient, 79
- cterasdk.client.host, 80
- cterasdk.client.http, 81
- cterasdk.client.ssl, 83
- cterasdk.common, 83
- cterasdk.common.datetime\_utils, 83
- cterasdk.common.item, 83
- cterasdk.common.object, 83
- cterasdk.common.types, 84
- cterasdk.config, 182
- cterasdk.convert, 88
- cterasdk.convert.exception, 88
- cterasdk.convert.format, 88
- cterasdk.convert.parse, 88
- cterasdk.convert.xml\_types, 89
- cterasdk.core, 89
- cterasdk.core.activation, 95
- cterasdk.core.antivirus, 95
- cterasdk.core.base\_command, 96
- cterasdk.core.buckets, 96
- cterasdk.core.cloudfs, 97
- cterasdk.core.connection, 99
- cterasdk.core.decorator, 99
- cterasdk.core.devices, 99
- cterasdk.core.directoryservice, 101
- cterasdk.core.enum, 103
- cterasdk.core.files, 89
- cterasdk.core.files.browser, 89
- cterasdk.core.files.collaboration, 92
- cterasdk.core.files.common, 93
- cterasdk.core.files.cp, 93
- cterasdk.core.files.directory, 93
- cterasdk.core.files.file\_access, 93
- cterasdk.core.files.ln, 94
- cterasdk.core.files.ls, 94
- cterasdk.core.files.mv, 94
- cterasdk.core.files.path, 94
- cterasdk.core.files.recover, 94
- cterasdk.core.files.rename, 94
- cterasdk.core.files.rm, 95
- cterasdk.core.login, 115
- cterasdk.core.logs, 115
- cterasdk.core.plans, 120
- cterasdk.core.portals, 116
- cterasdk.core.query, 118
- cterasdk.core.remote, 121
- cterasdk.core.reports, 119
- cterasdk.core.servers, 121
- cterasdk.core.session, 122
- cterasdk.core.setup, 122
- cterasdk.core.startup, 123
- cterasdk.core.syslog, 123
- cterasdk.core.taskmgr, 124
- cterasdk.core.templates, 125
- cterasdk.core.types, 127
- cterasdk.core.users, 131
- cterasdk.core.zones, 132
- cterasdk.edge, 133
- cterasdk.edge.afp, 136
- cterasdk.edge.aio, 136
- cterasdk.edge.array, 136
- cterasdk.edge.audit, 137
- cterasdk.edge.backup, 137
- cterasdk.edge.base\_command, 139
- cterasdk.edge.cache, 139
- cterasdk.edge.cli, 140
- cterasdk.edge.config, 140
- cterasdk.edge.connection, 141
- cterasdk.edge.decorator, 141
- cterasdk.edge.directoryservice, 141
- cterasdk.edge.drive, 142
- cterasdk.edge.enum, 143
- cterasdk.edge.files, 133
- cterasdk.edge.files.browser, 133
- cterasdk.edge.files.copy, 135
- cterasdk.edge.files.file\_access, 135
- cterasdk.edge.files.mkdir, 135
- cterasdk.edge.files.move, 135
- cterasdk.edge.files.path, 135
- cterasdk.edge.files.rm, 135
- cterasdk.edge.firmware, 153
- cterasdk.edge.ftp, 152
- cterasdk.edge.groups, 153
- cterasdk.edge.licenses, 154
- cterasdk.edge.login, 154
- cterasdk.edge.logs, 154
- cterasdk.edge.mail, 155
- cterasdk.edge.network, 155
- cterasdk.edge.nfs, 157
- cterasdk.edge.ntp, 157
- cterasdk.edge.power, 159
- cterasdk.edge.query, 159
- cterasdk.edge.remote, 160
- cterasdk.edge.rsync, 160
- cterasdk.edge.services, 160
- cterasdk.edge.session, 161
- cterasdk.edge.shares, 162
- cterasdk.edge.shell, 165
- cterasdk.edge.smb, 166
- cterasdk.edge.ssh, 158
- cterasdk.edge.ssl, 158
- cterasdk.edge.support, 167
- cterasdk.edge.sync, 168
- cterasdk.edge.syslog, 169
- cterasdk.edge.taskmgr, 170
- cterasdk.edge.telnet, 170
- cterasdk.edge.timezone, 170

- cterasdk.edge.types, 171
- cterasdk.edge.uri, 172
- cterasdk.edge.users, 173
- cterasdk.edge.volumes, 174
- cterasdk.exception, 182
- cterasdk.lib, 174
- cterasdk.lib.cmd, 174
- cterasdk.lib.consent, 175
- cterasdk.lib.file\_access\_base, 175
- cterasdk.lib.filesystem, 175
- cterasdk.lib.iterator, 176
- cterasdk.lib.platform, 176
- cterasdk.lib.registry, 176
- cterasdk.lib.session\_base, 176
- cterasdk.lib.tempfile, 177
- cterasdk.lib.tracker, 177
- cterasdk.lib.version, 177
- cterasdk.object, 178
- cterasdk.object.Agent, 178
- cterasdk.object.Gateway, 178
- cterasdk.object.Portal, 180
- cterasdk.transcript, 182
- cterasdk.transcript.transcribe, 182
- Monthly (*cterasdk.core.enum.PlanRetention* attribute), 110
- more\_than() (*cterasdk.common.types.IntegerCriteriaBuilder* static method), 86
- MoreThanOperator (*class in cterasdk.common.types*), 86
- Mounting (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- move() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- move() (*cterasdk.client.host.CTERAHost* method), 80
- move() (*cterasdk.client.http.HTTPClient* method), 81
- move() (*cterasdk.core.files.browser.CloudDrive* method), 90
- move() (*cterasdk.edge.files.browser.FileBrowser* method), 134
- move() (*in module cterasdk.core.files.mv*), 94
- move() (*in module cterasdk.edge.files.move*), 135
- move\_multi() (*cterasdk.core.files.browser.CloudDrive* method), 90
- move\_multi() (*in module cterasdk.core.files.mv*), 94
- multipart() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- multipart() (*cterasdk.client.host.CTERAHost* method), 80
- multipart() (*cterasdk.client.http.HTTPClient* method), 81
- NA (*cterasdk.core.enum.FileAccessMode* attribute), 145
- Name (*cterasdk.core.enum.TemplateCriteria* attribute), 115
- name (*cterasdk.core.types.CloudFSFolderFindingHelper* property), 127
- name (*cterasdk.core.types.PlatformVersion* property), 128
- name (*cterasdk.edge.types.ShareAccessControlEntry* property), 171
- name() (*cterasdk.common.types.FileFilterBuilder* static method), 85
- name() (*cterasdk.core.files.path.CTERAPath* method), 94
- name() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130
- name() (*cterasdk.edge.files.path.CTERAPath* method), 135
- name\_attr (*cterasdk.core.devices.Devices* attribute), 101
- names() (*cterasdk.common.types.FileFilterBuilder* static method), 85
- ne() (*cterasdk.core.query.FilterBuilder* method), 118
- NetAppStorageGRID (*class in cterasdk.core.types*), 128
- NetAppStorageGRID (*cterasdk.core.enum.BucketType* attribute), 103
- NetAppStorageGRID (*cterasdk.core.enum.LocationType* attribute), 108
- netmask (*cterasdk.edge.types.NFSv3AccessControlEntry* property), 171
- netmask (*cterasdk.edge.types.RemoveNFSv3AccessControlEntry* property), 171
- Network (*class in cterasdk.edge.network*), 155
- NetworkHost (*class in cterasdk.client.host*), 81
- NFS (*class in cterasdk.edge.nfs*), 157
- NFSv3AccessControlEntry (*class in cterasdk.edge.types*), 171
- no\_access() (*cterasdk.core.types.ShareRecipient* method), 129
- NoFolder (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- NoFolder (*cterasdk.edge.enum.SyncStatus* attribute), 150
- NonDeleted (*cterasdk.core.enum.ListFilter* attribute), 108
- NONE (*cterasdk.core.enum.PolicyType* attribute), 111
- none (*cterasdk.edge.support.DebugLevel* attribute), 167
- NOT\_EQUALS (*cterasdk.core.query.Restriction* attribute), 119
- NotFound, 139
- NotFound (*cterasdk.edge.backup.AttachRC* attribute), 137
- NOTICE (*cterasdk.core.enum.Severity* attribute), 114
- NOTICE (*cterasdk.edge.enum.Severity* attribute), 149
- NotInitialized (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- NotInitialized (*cterasdk.edge.enum.SyncStatus* attribute), 144

## N

- NA (*cterasdk.core.enum.FileAccessMode* attribute), 107
- NA (*cterasdk.core.enum.SetupWizardStatus* attribute), 113

attribute), 150

notLike() (*cterasdk.core.query.FilterBuilder* method), 118

NS (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89

NT1 (*cterasdk.edge.enum.SMBProtocol* attribute), 147

NTP (class in *cterasdk.edge.ntp*), 157

ntp (*cterasdk.edge.support.DebugLevel* attribute), 167

Nutanix (class in *cterasdk.core.types*), 128

Nutanix (*cterasdk.core.enum.BucketType* attribute), 103

## O

OBJ (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89

Object (class in *cterasdk.common.object*), 83

ObjectNotFoundException, 183

obtain\_ticket() (in module *cterasdk.edge.remote*), 160

Off (*cterasdk.edge.enum.SyncStatus* attribute), 150

OK (*cterasdk.edge.backup.AttachRC* attribute), 138

OK (*cterasdk.edge.backup.CreateFolderRC* attribute), 138

Ok (*cterasdk.edge.enum.VolumeStatus* attribute), 151

on\_ssl\_error() (*cterasdk.client.http.HttpClientBase* method), 82

on\_timeout() (*cterasdk.client.http.HttpClientBase* static method), 82

OnlyAuthenticatedUsers (*cterasdk.edge.enum.Acl* attribute), 143

Open (*cterasdk.edge.enum.TCPConnectRC* attribute), 150

openfile() (*cterasdk.client.host.CTERAHost* method), 80

openfile() (*cterasdk.edge.files.browser.FileBrowser* method), 134

OperatingSystem (*cterasdk.core.enum.TemplateCriteria* attribute), 115

OperationMode (class in *cterasdk.edge.enum*), 146

Operator (class in *cterasdk.common.types*), 86

orFilter() (*cterasdk.core.query.QueryParamBuilder* method), 118

OriginType (class in *cterasdk.core.enum*), 109

os() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130

os() (*cterasdk.lib.platform.Platform* method), 176

OSX (*cterasdk.core.enum.Platform* attribute), 111

OutOfQuota (*cterasdk.edge.enum.SyncStatus* attribute), 150

ownedBy() (*cterasdk.core.query.QueryParamBuilder* method), 118

Owner (*cterasdk.core.enum.TemplateCriteria* attribute), 115

owner (*cterasdk.core.types.CloudFSFolderFindingHelper* property), 127

owner() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130

## P

parent() (*cterasdk.core.files.path.CTERAPath* method), 94

parent() (*cterasdk.edge.files.path.CTERAPath* method), 135

ParseException, 88

ParseValue() (in module *cterasdk.convert.parse*), 88

parts() (*cterasdk.core.files.path.CTERAPath* method), 94

parts() (*cterasdk.edge.files.path.CTERAPath* method), 135

Password (*cterasdk.core.enum.SlaveAuthenticaitonMethod* attribute), 114

path() (*cterasdk.common.types.FileFilterBuilder* static method), 85

paths() (*cterasdk.common.types.FileFilterBuilder* static method), 86

perm (*cterasdk.edge.types.NFSv3AccessControlEntry* property), 171

perm (*cterasdk.edge.types.ShareAccessControlEntry* property), 171

PermissionDenied (*cterasdk.edge.backup.AttachRC* attribute), 138

PermissionDenied (*cterasdk.edge.backup.CreateFolderRC* attribute), 138

pin() (*cterasdk.edge.cache.Cache* method), 139

pin\_all() (*cterasdk.edge.cache.Cache* method), 139

pin\_exclude() (*cterasdk.edge.cache.Cache* method), 139

Plan (*cterasdk.core.enum.TemplateCriteria* attribute), 115

plan() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130

PlanAutoAssignPolicy (class in *cterasdk.core.plans*), 120

PlanCriteria (class in *cterasdk.core.enum*), 109

PlanCriteriaBuilder (class in *cterasdk.core.types*), 128

PlanItem (class in *cterasdk.core.enum*), 109

PlanRetention (class in *cterasdk.core.enum*), 110

Plans (class in *cterasdk.core.plans*), 120

Platform (class in *cterasdk.core.enum*), 111

Platform (class in *cterasdk.lib.platform*), 176

platform (*cterasdk.core.types.TemplateScript* property), 130

PlatformVersion (class in *cterasdk.core.types*), 128

PO (*cterasdk.core.enum.FileAccessMode* attribute), 107

PolicyRule (class in *cterasdk.common.types*), 86

PolicyRuleConverter (class in *cterasdk.common.types*), 86

PolicyType (class in *cterasdk.core.enum*), 111

port (*cterasdk.edge.types.TCPConnectResult* property), 172

port (*cterasdk.edge.types.TCPService* property), 172



- port() (*cterasdk.client.host.NetworkHost* method), 81
- Portal (*class in cterasdk.object.Portal*), 180
- Portal (*cterasdk.core.enum.OriginType* attribute), 109
- Portal (*cterasdk.core.enum.SetupWizardStage* attribute), 113
- PortalAccount (*class in cterasdk.core.types*), 128
- PortalAccountType (*class in cterasdk.core.enum*), 111
- Portals (*class in cterasdk.core.portals*), 116
- portals() (*cterasdk.core.reports.Reports* method), 119
- PortalType (*class in cterasdk.core.enum*), 112
- post() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- post() (*cterasdk.client.host.CTERAHost* method), 80
- post() (*cterasdk.client.http.HTTPClient* method), 82
- Power (*class in cterasdk.edge.power*), 159
- preview\_only() (*cterasdk.core.types.ShareRecipient* method), 130
- primary (*cterasdk.core.types.DomainControllers* property), 127
- PRIMARYUSER (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- principal\_type (*cterasdk.edge.types.ShareAccessControlEntry* property), 172
- principal\_type (*cterasdk.edge.types.UserGroupEntry* property), 172
- PrincipalType (*class in cterasdk.edge.enum*), 146
- PrivateKey (*cterasdk.core.enum.SlaveAuthenticaitonMethod* attribute), 114
- process (*cterasdk.edge.support.DebugLevel* attribute), 167
- PROGRAMFILES (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- PROJECTS (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- ProtectionLevel (*class in cterasdk.core.enum*), 112
- Public (*cterasdk.core.enum.ProtectionLevel* attribute), 112
- public\_info() (*cterasdk.object.Portal.Portal* method), 181
- put() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- put() (*cterasdk.client.host.CTERAHost* method), 80
- put() (*cterasdk.client.http.HTTPClient* method), 82
- put() (*cterasdk.core.query.QueryParamBuilder* method), 118
- put() (*cterasdk.edge.query.QueryParamBuilder* method), 159
- put() (*cterasdk.exception.CTERAException* method), 182
- put() (*cterasdk.object.Portal.Portal* method), 181
- python\_version() (*cterasdk.lib.platform.Platform* method), 176
- PythonVersionException, 183
- Q**
- Quarterly (*cterasdk.core.enum.PlanRetention* attribute), 110
- query() (*cterasdk.object.Gateway.Gateway* method), 179
- query() (*cterasdk.object.Portal.Portal* method), 181
- query() (*in module cterasdk.core.query*), 119
- query() (*in module cterasdk.edge.query*), 159
- QueryParam (*class in cterasdk.edge.query*), 159
- QueryParamBuilder (*class in cterasdk.core.query*), 118
- QueryParamBuilder (*class in cterasdk.edge.query*), 159
- QueryParams (*class in cterasdk.core.query*), 119
- R**
- RAID\_0 (*cterasdk.edge.enum.RAIDLevel* attribute), 147
- RAID\_1 (*cterasdk.edge.enum.RAIDLevel* attribute), 147
- RAID\_5 (*cterasdk.edge.enum.RAIDLevel* attribute), 147
- RAID\_6 (*cterasdk.edge.enum.RAIDLevel* attribute), 147
- RAIDLevel (*class in cterasdk.edge.enum*), 147
- read() (*cterasdk.client.http.HTTPResponse* method), 82
- read\_only() (*cterasdk.core.buckets.Buckets* method), 97
- read\_only() (*cterasdk.core.types.ShareRecipient* method), 130
- read\_write() (*cterasdk.core.buckets.Buckets* method), 97
- read\_write() (*cterasdk.core.types.ShareRecipient* method), 130
- ReadExtendedAttributes (*cterasdk.edge.enum.AuditEvents* attribute), 143
- ReadOnly (*cterasdk.edge.enum.VolumeStatus* attribute), 151
- ReadOnlyAdmin (*cterasdk.core.enum.Role* attribute), 112
- ReadOnlyAdministrators (*cterasdk.edge.enum.LocalGroup* attribute), 146
- ReadWriteAdmin (*cterasdk.core.enum.Role* attribute), 112
- reboot() (*cterasdk.edge.power.Power* method), 159
- reconnect() (*cterasdk.edge.services.Services* method), 161
- RECORDED\_HEADERS (*cterasdk.transcript.transcribe.Transcribe* attribute), 182
- Recoverable (*cterasdk.edge.backup.EncryptionMode* attribute), 138
- Recovering (*cterasdk.edge.enum.VolumeStatus* attribute), 152
- ref() (*cterasdk.core.query.FilterBuilder* static method), 118
- RefFilter (*cterasdk.core.query.FilterType* attribute), 118
- refresh() (*cterasdk.edge.sync.Sync* method), 168

- register() (*cterasdk.lib.registry.Registry* method), 176
- register\_session() (*cterasdk.client.host.CTERAHost* method), 80
- Registry (class in *cterasdk.lib.registry*), 176
- RejectedByPolicy (*cterasdk.edge.enum.SyncStatus* attribute), 150
- Remote (*cterasdk.edge.session.SessionType* attribute), 162
- remote() (*cterasdk.edge.session.Session* method), 161
- remote() (in module *cterasdk.edge.uri*), 172
- remote\_access() (*cterasdk.edge.session.Session* method), 161
- remote\_access() (*cterasdk.object.Gateway.Gateway* method), 179
- remote\_access() (in module *cterasdk.edge.remote*), 160
- remote\_access() (in module *cterasdk.edge.uri*), 172
- remote\_command() (in module *cterasdk.core.remote*), 121
- remote\_from() (*cterasdk.edge.session.Session* method), 161
- RemoteDirectoryNotFound, 183
- RemoteFileSystemException, 183
- remove() (*cterasdk.lib.registry.Registry* method), 176
- remove\_acl() (*cterasdk.edge.shares.Shares* method), 164
- remove\_array\_element() (in module *cterasdk.common.object*), 84
- remove\_array\_element\_by\_key() (in module *cterasdk.common.object*), 84
- remove\_attr() (in module *cterasdk.common.object*), 84
- remove\_default() (*cterasdk.core.templates.Templates* method), 126
- remove\_file\_exclusion\_rules() (*cterasdk.edge.sync.Sync* method), 169
- remove\_members() (*cterasdk.edge.groups.Groups* method), 153
- remove\_pin() (*cterasdk.edge.cache.Cache* method), 139
- remove\_screened\_file\_types() (*cterasdk.edge.shares.Shares* method), 164
- remove\_share\_recipients() (*cterasdk.core.files.browser.CloudDrive* method), 90
- remove\_share\_recipients() (in module *cterasdk.core.files.collaboration*), 92
- remove\_static\_domain\_controller() (*cterasdk.edge.directoryservice.DirectoryService* method), 142
- remove\_storage\_ca() (*cterasdk.edge.ssl.SSL* method), 158
- remove\_trusted\_nfs\_clients() (*cterasdk.edge.shares.Shares* method), 164
- RemoveNFSv3AccessControlEntry (class in *cterasdk.edge.types*), 171
- RemoveShareAccessControlEntry (class in *cterasdk.edge.types*), 171
- rename() (*cterasdk.core.files.browser.CloudDrive* method), 90
- rename() (*cterasdk.lib.filesystem.FileSystem* method), 175
- rename() (in module *cterasdk.core.files.rename*), 94
- RenameException, 183
- Repairing (*cterasdk.edge.enum.VolumeStatus* attribute), 152
- Replication (*cterasdk.core.enum.SetupWizardStage* attribute), 113
- Reports (class in *cterasdk.core.reports*), 119
- Required (*cterasdk.edge.enum.CIFSPacketSigning* attribute), 144
- rescan() (*cterasdk.core.antivirus.Antivirus* method), 95
- Reseller (*cterasdk.core.enum.PortalType* attribute), 112
- ReservedName, 93
- reset() (*cterasdk.edge.power.Power* method), 159
- reset\_mtu() (*cterasdk.edge.network.Network* method), 156
- Resizing (*cterasdk.edge.enum.VolumeStatus* attribute), 152
- resolve() (*cterasdk.lib.tracker.StatusTracker* method), 177
- ResolvingServers (*cterasdk.edge.enum.ServicesConnectionState* attribute), 148
- Restart (*cterasdk.core.enum.SetupWizardStage* attribute), 113
- restart() (*cterasdk.edge.smb.SMB* method), 166
- Restriction (class in *cterasdk.core.query*), 119
- rm() (*cterasdk.object.Gateway.Gateway* method), 179
- rmdir() (*cterasdk.lib.tempfile.TempfileServices* static method), 177
- rmfg() (*cterasdk.core.cloudfs.CloudFS* method), 99
- RO (*cterasdk.core.enum.FileAccessMode* attribute), 107
- RO (*cterasdk.edge.enum.FileAccessMode* attribute), 145
- Role (class in *cterasdk.core.enum*), 112
- Role (*cterasdk.core.enum.PlanCriteria* attribute), 109
- role (*cterasdk.core.types.AccessControlEntry* property), 127
- role() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128
- root() (*cterasdk.common.types.DirectoryEntryFactory* static method), 85
- RSync (class in *cterasdk.edge.rsync*), 160
- rsync (*cterasdk.edge.support.DebugLevel* attribute), 167
- run\_command() (*cterasdk.edge.cli.CLI* method), 140
- run\_command() (*cterasdk.edge.shell.Shell* method), 165
- Running (*cterasdk.core.enum.SetupWizardStatus* attribute), 114
- Running (*cterasdk.edge.enum.TaskStatus* attribute), 150
- running() (*cterasdk.edge.taskmgr.Tasks* method), 170

- `running()` (*cterasdk.lib.tracker.StatusTracker* method), 177
- `RW` (*cterasdk.core.enum.FileAccessMode* attribute), 107
- `RW` (*cterasdk.edge.enum.FileAccessMode* attribute), 145
- ## S
- `S3` (*cterasdk.core.enum.LocationType* attribute), 108
- `S3Compatible` (class in *cterasdk.core.types*), 129
- `S3Compatible` (*cterasdk.core.enum.BucketType* attribute), 103
- `S3Compatible` (*cterasdk.core.enum.LocationType* attribute), 108
- `SA` (*cterasdk.core.enum.PlanItem* attribute), 110
- `samba` (*cterasdk.edge.support.DebugLevel* attribute), 167
- `save()` (*cterasdk.lib.filesystem.FileSystem* method), 175
- `save_cert_from_server()` (*cterasdk.client.ssl.CertificateServices* static method), 83
- `Scality` (class in *cterasdk.core.types*), 129
- `Scality` (*cterasdk.core.enum.BucketType* attribute), 104
- `Scanning` (*cterasdk.edge.enum.SyncStatus* attribute), 150
- `schedule()` (*cterasdk.common.types.ThrottlingRuleBuilder* method), 87
- `scheme()` (*cterasdk.client.host.NetworkHost* method), 81
- `SearchType` (class in *cterasdk.core.enum*), 112
- `secondary` (*cterasdk.core.types.DomainControllers* property), 127
- `Secret` (*cterasdk.edge.backup.EncryptionMode* attribute), 138
- `SECTION_END` (*cterasdk.transcript.transcribe.Transcribe* attribute), 182
- `SECTION_START` (*cterasdk.transcript.transcribe.Transcribe* attribute), 182
- `SELECT` (*cterasdk.core.enum.PolicyType* attribute), 111
- `Server` (*cterasdk.core.enum.SetupWizardStage* attribute), 113
- `ServerAgent` (*cterasdk.core.enum.DeviceType* attribute), 105
- `ServerMode` (class in *cterasdk.core.enum*), 113
- `Servers` (class in *cterasdk.core.servers*), 121
- `servers` (*cterasdk.edge.ntp.NTP* property), 157
- `servers()` (*cterasdk.core.devices.Devices* method), 101
- `Services` (class in *cterasdk.edge.services*), 160
- `ServicesConnectionState` (class in *cterasdk.edge.enum*), 148
- `ServicesPortal` (class in *cterasdk.object.Portal*), 181
- `ServicesPortal` (*cterasdk.core.enum.Context* attribute), 104
- `ServiceUnavailable` (*cterasdk.edge.enum.SyncStatus* attribute), 150
- `Session` (class in *cterasdk.core.session*), 122
- `Session` (class in *cterasdk.edge.session*), 161
- `session()` (*cterasdk.client.host.CTERAHost* method), 81
- `session()` (*cterasdk.core.base\_command.BaseCommand* method), 96
- `session()` (*cterasdk.edge.base\_command.BaseCommand* method), 139
- `SessionBase` (class in *cterasdk.lib.session\_base*), 176
- `SessionConnection` (class in *cterasdk.edge.session*), 161
- `SessionStatus` (class in *cterasdk.lib.session\_base*), 176
- `SessionType` (class in *cterasdk.edge.session*), 161
- `SessionUser` (class in *cterasdk.lib.session\_base*), 177
- `set_access_control()` (*cterasdk.core.directoryservice.DirectoryService* method), 102
- `set_access_type()` (*cterasdk.edge.shares.Shares* method), 164
- `set_acl()` (*cterasdk.edge.shares.Shares* method), 164
- `set_advanced_mapping()` (*cterasdk.core.directoryservice.DirectoryService* method), 102
- `set_authorization_headers()` (*cterasdk.client.cteraclient.CTERAClient* method), 79
- `set_authorization_headers()` (*cterasdk.client.host.CTERAHost* method), 81
- `set_custom_headers()` (*cterasdk.client.http.HttpClientBase* method), 82
- `set_debug_level()` (*cterasdk.edge.support.Support* method), 168
- `set_default()` (*cterasdk.core.templates.Templates* method), 126
- `set_hostname()` (*cterasdk.edge.config.Config* method), 140
- `set_linux_avoid_using_fanotify()` (*cterasdk.edge.sync.Sync* method), 169
- `set_location()` (*cterasdk.edge.config.Config* method), 141
- `set_mtu()` (*cterasdk.edge.network.Network* method), 156
- `set_packet_signing()` (*cterasdk.edge.smb.SMB* method), 166
- `set_policy()` (*cterasdk.core.plans.PlanAutoAssignPolicy* method), 120
- `set_policy()` (*cterasdk.core.templates.TemplateAutoAssignPolicy* method), 125
- `set_policy()` (*cterasdk.edge.sync.CloudSyncBandwidthThrottling* method), 168
- `set_screened_file_types()` (*cterasdk.edge.shares.Shares* method), 165
- `set_session_id()` (*cterasdk.client.cteraclient.CTERAClient* method), 79

- set\_session\_id() (*cterasdk.client.host.CTERAHost* method), 81  
 set\_session\_id() (*cterasdk.client.http.HttpClientBase* method), 82  
 set\_share\_winacls() (*cterasdk.edge.shares.Shares* method), 165  
 set\_static\_domain\_controller() (*cterasdk.edge.directoryservice.DirectoryService* method), 142  
 set\_static\_ipaddr() (*cterasdk.edge.network.Network* method), 156  
 set\_static\_nameserver() (*cterasdk.edge.network.Network* method), 156  
 set\_timezone() (*cterasdk.edge.timezone.Timezone* method), 170  
 set\_trusted\_nfs\_clients() (*cterasdk.edge.shares.Shares* method), 165  
 SetAppendValue() (in module *cterasdk.convert.parse*), 88  
 setLevel() (*cterasdk.config.Logging* static method), 182  
 settings() (*cterasdk.edge.logs.Logs* method), 155  
 Setup (class in *cterasdk.core.setup*), 122  
 SetupWizardStage (class in *cterasdk.core.enum*), 113  
 SetupWizardStatus (class in *cterasdk.core.enum*), 113  
 SetupWizardStatusMonitor (class in *cterasdk.core.setup*), 123  
 setValue() (*cterasdk.core.query.FilterBuilder* method), 118  
 Severity (class in *cterasdk.core.enum*), 114  
 Severity (class in *cterasdk.edge.enum*), 148  
 Share (*cterasdk.core.enum.PlanItem* attribute), 110  
 share() (*cterasdk.core.files.brower.CloudDrive* method), 90  
 share() (in module *cterasdk.core.files.collaboration*), 92  
 ShareAccessControlEntry (class in *cterasdk.edge.types*), 171  
 ShareRecipient (class in *cterasdk.core.types*), 129  
 Shares (class in *cterasdk.edge.shares*), 162  
 Shell (class in *cterasdk.edge.shell*), 165  
 should\_trust() (*cterasdk.client.http.HttpClientBase* method), 82  
 ShouldSupportWinNtAcl (*cterasdk.edge.enum.SyncStatus* attribute), 150  
 show() (*cterasdk.client.host.CTERAHost* method), 81  
 show() (in module *cterasdk.core.query*), 119  
 show() (in module *cterasdk.edge.query*), 159  
 show\_multi() (*cterasdk.client.host.CTERAHost* method), 81  
 show\_query() (*cterasdk.object.Gateway.Gateway* method), 180  
 show\_query() (*cterasdk.object.Portal.Portal* method), 181  
 shutdown() (*cterasdk.edge.power.Power* method), 159  
 size() (*cterasdk.common.types.FileFilterBuilder* static method), 86  
 Slave (*cterasdk.core.enum.ServerMode* attribute), 113  
 SlaveAuthenticaiionMethod (class in *cterasdk.core.enum*), 114  
 SMB (class in *cterasdk.edge.smb*), 166  
 SMB1 (*cterasdk.edge.enum.SMBProtocol* attribute), 147  
 SMB2 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB2\_02 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB2\_10 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB3 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB3\_00 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB3\_02 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMB3\_11 (*cterasdk.edge.enum.SMBProtocol* attribute), 148  
 SMBProtocol (class in *cterasdk.edge.enum*), 147  
 Sophos (*cterasdk.core.enum.AntivirusType* attribute), 103  
 sortBy() (*cterasdk.core.query.QueryParamBuilder* method), 119  
 split\_file\_directory() (*cterasdk.lib.filesystem.FileSystem* static method), 175  
 split\_file\_directory\_with\_defaults() (*cterasdk.lib.filesystem.FileSystem* static method), 175  
 SrcDstParam (class in *cterasdk.core.files.common*), 93  
 SSH (class in *cterasdk.edge.ssh*), 158  
 SSL (class in *cterasdk.edge.ssl*), 158  
 SSLException, 183  
 sso\_enabled() (*cterasdk.edge.services.Services* method), 161  
 start() (*cterasdk.common.types.TimeRange* method), 87  
 start() (*cterasdk.edge.backup.Backup* method), 138  
 start\_local\_session() (*cterasdk.lib.session\_base.SessionBase* method), 176  
 start\_remote\_session() (*cterasdk.edge.session.Session* method), 161  
 Started (*cterasdk.core.startup.Startup* attribute), 123  
 startFrom() (*cterasdk.core.query.QueryParamBuilder* method), 119  
 startFrom() (*cterasdk.edge.query.QueryParamBuilder* method), 159  
 startswith() (*cterasdk.common.types.StringCriteriaBuilder* method), 86



- Startup (class in *cterasdk.core.startup*), 123
- status() (*cterasdk.core.antivirus.Antivirus* method), 95
- status() (*cterasdk.core.startup.Startup* method), 123
- status() (*cterasdk.core.taskmgr.Tasks* method), 124
- status() (*cterasdk.edge.taskmgr.Tasks* method), 170
- StatusTracker (class in *cterasdk.lib.tracker*), 177
- Storage (*cterasdk.core.enum.PlanItem* attribute), 110
- storage (*cterasdk.edge.support.DebugLevel* attribute), 167
- storage() (*cterasdk.core.reports.Reports* method), 119
- String (*cterasdk.core.query.FilterType* attribute), 118
- StringCriteriaBuilder (class in *cterasdk.common.types*), 86
- subscribe() (*cterasdk.core.portals.Portals* method), 117
- successful() (*cterasdk.lib.tracker.StatusTracker* method), 177
- Support (class in *cterasdk.edge.support*), 167
- Support (*cterasdk.core.enum.Role* attribute), 112
- suspend() (*cterasdk.core.antivirus.Antivirus* method), 95
- suspend() (*cterasdk.core.antivirus.AntivirusServers* method), 96
- suspend() (*cterasdk.edge.backup.Backup* method), 138
- suspend() (*cterasdk.edge.sync.Sync* method), 169
- Symantec (*cterasdk.core.enum.AntivirusType* attribute), 103
- Sync (class in *cterasdk.edge.sync*), 168
- Synced (*cterasdk.edge.enum.SyncStatus* attribute), 150
- Syncing (*cterasdk.edge.enum.SyncStatus* attribute), 150
- SYNCS (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- SyncStatus (class in *cterasdk.edge.enum*), 149
- Syslog (class in *cterasdk.core.syslog*), 123
- Syslog (class in *cterasdk.edge.syslog*), 169
- System (*cterasdk.core.enum.LogTopic* attribute), 108
- SYSTEMDRIVE (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- ## T
- TakingSnapshot (*cterasdk.edge.enum.SyncStatus* attribute), 150
- Task (class in *cterasdk.core.taskmgr*), 124
- Task (class in *cterasdk.edge.taskmgr*), 170
- Tasks (class in *cterasdk.core.taskmgr*), 124
- Tasks (class in *cterasdk.edge.taskmgr*), 170
- TaskSchedule (class in *cterasdk.common.types*), 86
- TaskStatus (class in *cterasdk.edge.enum*), 150
- TCP (*cterasdk.core.enum.IPProtocol* attribute), 107
- TCP (*cterasdk.edge.enum.IPProtocol* attribute), 145
- tcp\_connect() (*cterasdk.edge.network.Network* method), 157
- TCPConnectRC (class in *cterasdk.edge.enum*), 150
- TCPConnectResult (class in *cterasdk.edge.types*), 172
- TCPService (class in *cterasdk.edge.types*), 172
- Team (*cterasdk.core.enum.PortalType* attribute), 112
- Telnet (class in *cterasdk.edge.telnet*), 170
- TEMP (*cterasdk.core.enum.EnvironmentVariables* attribute), 106
- TempfileServices (class in *cterasdk.lib.tempfile*), 177
- TemplateAutoAssignPolicy (class in *cterasdk.core.templates*), 125
- TemplateCriteria (class in *cterasdk.core.enum*), 114
- TemplateCriteriaBuilder (class in *cterasdk.core.types*), 130
- Templates (class in *cterasdk.core.templates*), 125
- TemplateScript (class in *cterasdk.core.types*), 130
- tenant() (*cterasdk.lib.session\_base.SessionBase* method), 176
- tenants() (*cterasdk.core.portals.Portals* method), 117
- terminate() (*cterasdk.lib.session\_base.SessionBase* method), 176
- terminate\_at\_endtime() (*cterasdk.common.types.TimeRange* method), 87
- test() (*cterasdk.object.Gateway.Gateway* method), 180
- test() (*cterasdk.object.Portal.Portal* method), 181
- test() (in module *cterasdk.core.connection*), 99
- test() (in module *cterasdk.edge.connection*), 141
- test\_application() (in module *cterasdk.edge.connection*), 141
- test\_conn() (*cterasdk.client.host.NetworkHost* method), 81
- test\_network() (in module *cterasdk.core.connection*), 99
- test\_network() (in module *cterasdk.edge.connection*), 141
- textplain (*cterasdk.client.http.ContentType* attribute), 81
- ThrottlingRule (class in *cterasdk.common.types*), 86
- ThrottlingRuleBuilder (class in *cterasdk.common.types*), 87
- TimeRange (class in *cterasdk.common.types*), 87
- Timezone (class in *cterasdk.edge.timezone*), 170
- to\_server\_object() (*cterasdk.common.types.ThrottlingRule* method), 87
- to\_server\_object() (*cterasdk.core.types.AmazonS3* method), 127
- to\_server\_object() (*cterasdk.core.types.AzureBlob* method), 127
- to\_server\_object() (*cterasdk.core.types.Bucket* method), 127
- to\_server\_object() (*cterasdk.core.types.NetAppStorageGRID* method), 128
- to\_server\_object() (*cterasdk.core.types.S3Compatible* method), 129
- to\_server\_object() (*cterasdk.core.types.TemplateScript* method), 130

- to\_server\_object() (*cterasdk.edge.types.NFSv3AccessControlEntry* attribute), 171
- to\_server\_object() (*cterasdk.edge.types.ShareAccessControlEntry* attribute), 172
- to\_server\_object() (*cterasdk.edge.types.UserGroupEntry* attribute), 172
- tojsonstr() (in module *cterasdk.convert.format*), 88
- toxml() (in module *cterasdk.convert.format*), 88
- toxmlstr() (in module *cterasdk.convert.format*), 88
- track() (*cterasdk.lib.tracker.StatusTracker* method), 177
- track() (in module *cterasdk.lib.tracker*), 177
- Traffic (class in *cterasdk.edge.enum*), 151
- Transcribe (class in *cterasdk.transcript.transcribe*), 182
- transcribe() (*cterasdk.transcript.transcribe.Transcribe* method), 182
- transcribe() (in module *cterasdk.transcript.transcribe*), 182
- TraverseFolderExecuteFile (*cterasdk.edge.enum.AuditEvents* attribute), 143
- TrendMicro (*cterasdk.core.enum.AntivirusType* attribute), 103
- trust() (*cterasdk.client.http.HttpClientBase* method), 82
- Type (*cterasdk.common.types.FileFilterBuilder* attribute), 85
- Type (*cterasdk.core.enum.TemplateCriteria* attribute), 115
- Type (*cterasdk.core.types.PlanCriteriaBuilder* attribute), 128
- Type (*cterasdk.core.types.TemplateCriteriaBuilder* attribute), 130
- type() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130
- type\_attr (*cterasdk.core.devices.Devices* attribute), 101
- ## U
- UDP (*cterasdk.core.enum.IPProtocol* attribute), 107
- UDP (*cterasdk.edge.enum.IPProtocol* attribute), 145
- undeleate() (*cterasdk.core.cloudfs.CloudFS* method), 99
- undeleate() (*cterasdk.core.files.browser.CloudDrive* method), 91
- undeleate() (*cterasdk.core.portals.Portals* method), 117
- undeleate() (in module *cterasdk.core.files.recover*), 94
- undeleate\_multi() (*cterasdk.core.files.browser.CloudDrive* method), 91
- undeleate\_multi() (in module *cterasdk.core.files.recover*), 94
- Unknown (*cterasdk.edge.enum.VolumeStatus* attribute), 152
- Unlicensed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- Unlicensed (*cterasdk.edge.enum.SyncStatus* attribute), 150
- UnMounted (*cterasdk.edge.enum.VolumeStatus* attribute), 152
- unpin\_all() (*cterasdk.edge.cache.Cache* method), 139
- unselect\_all() (*cterasdk.edge.backup.BackupFiles* method), 138
- unshare() (*cterasdk.core.files.browser.CloudDrive* method), 91
- unshare() (in module *cterasdk.core.files.collaboration*), 92
- Unsubscribed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- unsuspend() (*cterasdk.core.antivirus.Antivirus* method), 95
- unsuspend() (*cterasdk.core.antivirus.AntivirusServers* method), 96
- unsuspend() (*cterasdk.edge.backup.Backup* method), 138
- unsuspend() (*cterasdk.edge.sync.Sync* method), 169
- update\_current\_tenant() (in module *cterasdk.core.decorator*), 99
- update\_tenant() (*cterasdk.core.session.Session* method), 122
- upgrade() (*cterasdk.edge.firmware.Firmware* method), 153
- UpgradingDataBase (*cterasdk.edge.enum.SyncStatus* attribute), 150
- Upload (*cterasdk.edge.enum.Traffic* attribute), 151
- upload (*cterasdk.edge.support.DebugLevel* attribute), 167
- upload() (*cterasdk.client.cteraclient.CTERAClient* method), 79
- upload() (*cterasdk.client.host.CTERAHost* method), 81
- upload() (*cterasdk.client.http.HTTPClient* method), 82
- upload() (*cterasdk.common.types.ThrottlingRuleBuilder* method), 87
- upload() (*cterasdk.core.files.browser.CloudDrive* method), 41, 91
- upload() (*cterasdk.edge.files.browser.FileBrowser* method), 134
- upload() (*cterasdk.lib.file\_access\_base.FileAccessBase* method), 175
- UploadTaskStatus (class in *cterasdk.edge.firmware*), 153
- urlencoded (*cterasdk.client.http.ContentType* attribute), 81
- User (*cterasdk.core.enum.DirectorySearchEntityType* attribute), 105
- User (*cterasdk.core.enum.PortalAccountType* attribute), 112
- user\_groups() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128

- UserAccount (class in *cterasdk.core.types*), 130  
 UserGroupEntry (class in *cterasdk.edge.types*), 172  
 Username (*cterasdk.core.enum.PlanCriteria* attribute), 109  
 username() (*cterasdk.core.types.PlanCriteriaBuilder* static method), 128  
 USERPROFILE (*cterasdk.core.enum.EnvironmentVariables* attribute), 106  
 Users (class in *cterasdk.core.users*), 131  
 Users (class in *cterasdk.edge.users*), 173  
 USERS (*cterasdk.core.enum.EnvironmentVariables* attribute), 107  
 Users (*cterasdk.core.enum.SearchType* attribute), 113  
 UUID (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89
- ## V
- VAL (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89  
 validate\_directory() (*cterasdk.lib.filesystem.FileSystem* method), 175  
 validate\_permission() (*cterasdk.edge.types.AccessControlEntryValidator* static method), 171  
 Version (class in *cterasdk.lib.version*), 177  
 VERSION (*cterasdk.convert.xml\_types.XMLTypes* attribute), 89  
 Version (*cterasdk.core.enum.TemplateCriteria* attribute), 115  
 version (*cterasdk.core.types.PlatformVersion* property), 128  
 version() (*cterasdk.core.types.TemplateCriteriaBuilder* static method), 130  
 version() (*cterasdk.lib.filesystem.FileSystem* static method), 175  
 vGateway (*cterasdk.core.enum.DeviceType* attribute), 105  
 Volumes (class in *cterasdk.edge.volumes*), 174  
 VolumeStatus (class in *cterasdk.edge.enum*), 151  
 VolumeUnavailable (*cterasdk.edge.enum.SyncStatus* attribute), 150
- ## W
- WA (*cterasdk.core.enum.PlanItem* attribute), 110  
 wait() (*cterasdk.core.setup.SetupWizardStatusMonitor* method), 123  
 wait() (*cterasdk.core.startup.Startup* method), 123  
 wait() (*cterasdk.core.taskmgr.Tasks* method), 124  
 wait() (*cterasdk.edge.power.Boot* method), 159  
 wait() (*cterasdk.edge.taskmgr.Tasks* method), 170  
 walk() (*cterasdk.core.files.browser.FileBrowser* method), 92  
 WARNING (*cterasdk.core.enum.Severity* attribute), 114  
 WARNING (*cterasdk.edge.enum.Severity* attribute), 149  
 warning (*cterasdk.edge.support.DebugLevel* attribute), 167  
 Wasabi (class in *cterasdk.core.types*), 131  
 Wasabi (*cterasdk.core.enum.BucketType* attribute), 104  
 Weekly (*cterasdk.core.enum.PlanRetention* attribute), 110  
 whoami() (*cterasdk.client.host.CTERAHost* method), 81  
 whoami() (*cterasdk.lib.session\_base.SessionBase* method), 176  
 WINDIR (*cterasdk.core.enum.EnvironmentVariables* attribute), 107  
 window() (*cterasdk.common.types.BackupScheduleBuilder* static method), 85  
 Windows (*cterasdk.core.enum.Platform* attribute), 111  
 windows() (*cterasdk.core.types.TemplateScript* static method), 130  
 WindowsNT (*cterasdk.edge.enum.Acl* attribute), 143  
 WorkstationAgent (*cterasdk.core.enum.DeviceType* attribute), 105  
 write() (*cterasdk.lib.filesystem.FileSystem* static method), 175  
 WriteAttributes (*cterasdk.edge.enum.AuditEvents* attribute), 143  
 WriteExtendedAttributes (*cterasdk.edge.enum.AuditEvents* attribute), 143  
 WrongPassword (*cterasdk.edge.enum.BackupConfStatusID* attribute), 144
- ## X
- XMLTypes (class in *cterasdk.convert.xml\_types*), 89
- ## Y
- Yearly (*cterasdk.core.enum.PlanRetention* attribute), 110
- ## Z
- Zones (class in *cterasdk.core.zones*), 132