

---

# **CTERA SDK for Python Documentation**

**CTERA Networks**

**Jun 12, 2020**



---

## Contents:

---

<b>1</b>	<b>CTERA for Python</b>	<b>1</b>
1.1	Documentation . . . . .	1
1.2	Installation . . . . .	1
1.3	Importing the Library . . . . .	2
1.4	Building Documentation . . . . .	2
1.5	Testing . . . . .	2
<b>2</b>	<b>User Guides</b>	<b>3</b>
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	Logging in . . . . .	4
2.1.1.3	Navigating . . . . .	5
2.1.1.4	Core Methods . . . . .	5
2.1.1.5	Portals . . . . .	6
2.1.1.6	Servers . . . . .	7
2.1.1.7	Users . . . . .	8
2.1.1.8	Devices . . . . .	10
2.1.1.9	Zones . . . . .	13
2.1.1.10	CloudFS . . . . .	15
2.1.2	End User Portal . . . . .	17
2.1.2.1	Instantiate a Services Portal object . . . . .	17
2.1.3	File Browser . . . . .	18
2.1.3.1	List . . . . .	19
2.1.3.2	Download . . . . .	19
2.1.3.3	Create Directory . . . . .	19
2.1.3.4	Rename . . . . .	19
2.1.3.5	Delete . . . . .	20
2.1.3.6	Recover . . . . .	20
2.1.3.7	Copy . . . . .	20
2.1.3.8	Move . . . . .	21
2.1.3.9	Create Public Link . . . . .	21
2.1.3.10	Collaboration Shares . . . . .	21
2.2	Edge Filer . . . . .	23
2.2.1	Gateway . . . . .	23
2.2.1.1	Instantiate a Gateway object . . . . .	24

2.2.2	File Browser	45
2.2.2.1	Obtaining Access to the Gateway's File System	46
2.3	Agent	47
2.4	Miscellaneous	47
2.4.1	Logging	47
2.4.1.1	Redirecting the log to a file	47
2.4.1.2	Disabling the Logger	47
2.4.1.3	Changing the Log Level	48
2.4.2	Formatting	48
<b>3</b>	<b>cterasdk package</b>	<b>51</b>
3.1	Subpackages	51
3.1.1	cterasdk.client package	51
3.1.1.1	Submodules	51
3.1.2	cterasdk.common package	54
3.1.2.1	Submodules	54
3.1.3	cterasdk.convert package	55
3.1.3.1	Submodules	55
3.1.4	cterasdk.core package	56
3.1.4.1	Subpackages	56
3.1.4.2	Submodules	61
3.1.5	cterasdk.edge package	78
3.1.5.1	Subpackages	78
3.1.5.2	Submodules	80
3.1.6	cterasdk.lib package	111
3.1.6.1	Submodules	111
3.1.7	cterasdk.object package	114
3.1.7.1	Submodules	114
3.1.8	cterasdk.transcript package	117
3.1.8.1	Submodules	117
3.2	Submodules	117
3.2.1	cterasdk.config module	117
3.2.2	cterasdk.exception module	117
<b>4</b>	<b>Indices and tables</b>	<b>119</b>
<b>5</b>	<b>Help Us Improve the Docs &lt;3</b>	<b>121</b>
	<b>Python Module Index</b>	<b>123</b>
	<b>Index</b>	<b>125</b>

# CHAPTER 1

---

## 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
```

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*
  - *Logging in*
  - *Navigating*
  - *Core Methods*
  - *Portals*
    - \* *Retrieve Portals*
    - \* *Create a Team Portal*
    - \* *Subscribe a Team Portal to a Plan*
    - \* *Delete a Team Portal*
    - \* *Recover a Team Portal*
  - *Servers*
  - *Users*
    - \* *Local Users*
    - \* *Domain Users*
    - \* *Fetch Users & Groups*

- *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*

### 2.1.1.1 Instantiate a Global Admin object

**class** `cterasdk.object.Portal.GlobalAdmin` (*host, port=443, 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

**`__init__`** (*host, port=443, 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), defaults to 443
- **`https`** (*bool, optional*) – Set to True to require HTTPS, defaults to True

```
admin = GlobalAdmin('chopin.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'`

### 2.1.1.2 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 str** The name of the logged in user

```
admin.whoami ()
```

### 2.1.1.3 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('chopin')
```

### 2.1.1.4 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*)

### 2.1.1.5 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')
```

(continues on next page)

(continued from previous page)

```

"""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')
```

## 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')
```

### 2.1.1.6 Servers

`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'])

for server in servers:

    print(server)

```

### 2.1.1.7 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:

```

(continues on next page)

(continued from previous page)

```
print (user)
```

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

Add a portal user

#### 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 an end user"""
```

```
admin.users.add('bruce', 'bruce.wayne@we.com', 'Bruce', 'Wayne', 'G0th4amCity!')
```

## 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'])
```

(continues on next page)

(continued from previous page)

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

admin.directoryservice.fetch([alice, bruce])
```

### 2.1.1.8 Devices

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

Get a Device by its name

**Parameters**

- **device\_name** (*str*) – Name of the device to retrieve
- **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()

for filer in filers:
```

(continues on next page)

(continued from previous page)

```

    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  
  
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 ...
```

### 2.1.1.9 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')
```

### 2.1.1.10 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*)

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)

```
"""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, deleted=False, user=None*)

List cloud drive folders

#### Parameters

- **include** (*str, optional*) – List of fields to retrieve, defaults to ['name', 'group', 'owner']
- **deleted** (*str, optional*) – Retrieve 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)
```

CloudFS.**mkdir** (*name, group, owner, winacls=True*)

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

```
"""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)

```

## 2.1.2 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=443, https=True*)

Main class for Service operations on a Portal

`__init__` (*host, port=443, 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), defaults to 443
- **https** (*bool, optional*) – Set to True to require HTTPS, defaults to True

```
user = ServicesPortal('chopin.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('admin', 'G3neralZ0d!')
```

`ServicesPortal.logout()`  
Log out

```
user.logout()
```

### 2.1.3 File Browser

#### Table of Contents

- *File Browser*
  - *List*
  - *Download*
  - *Create Directory*
  - *Rename*
  - *Delete*
  - *Recover*
  - *Copy*
  - *Move*
  - *Create Public Link*
  - *Collaboration Shares*

### 2.1.3.1 List

`FileBrowser.ls(path)`

Execute ls on the provided path

**Parameters** `path` (*str*) – Path to execute ls on

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

`FileBrowser.walk(path)`

Perform walk on the provided path

**Parameters** `path` (*str*) – Path to perform walk on

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

### 2.1.3.2 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 file-name will be kept, defaults to the default directory

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

### 2.1.3.3 Create Directory

`FileBrowser.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)
```

### 2.1.3.4 Rename

`FileBrowser.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')
```

### 2.1.3.5 Delete

FileBrowser.**delete** (*path*)

Delete a file

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

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

FileBrowser.**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')
```

### 2.1.3.6 Recover

FileBrowser.**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')
```

FileBrowser.**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')
```

### 2.1.3.7 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')
```



### 2.1.3.8 Move

FileBrowser.**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')
```

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

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

### 2.1.3.9 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.

### 2.1.3.10 Collaboration Shares

FileBrowser.**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 = []

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()
```

(continues on next page)

(continued from previous page)

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

`FileBrowser.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

---

`FileBrowser.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]`

`FileBrowser.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')
```

## 2.2 Edge Filer

### 2.2.1 Gateway

#### Table of Contents

- *Gateway*
  - *Instantiate a Gateway object*
  - \* *Logging in*

- \* *Core Methods*
- \* *Device Configuration*
- \* *Storage*
  - *Format*
  - *Volumes*
- \* *Shares*
- \* *Users*
- \* *Groups*
- \* *Active Directory*
- \* *Cloud Services*
- \* *Applying a License*
- \* *Caching*
- \* *Cloud Backup*
- \* *Cloud Sync*
- \* *File Access Protocols*
  - *Windows File Sharing (CIFS/SMB)*
- \* *Network*
  - *Network Diagnostics*
- \* *Mail Server*
- \* *Logging*
  - *SMB Audit Logs*
- \* *Reset*
- \* *SSL*
- \* *Power Management*
- \* *Support*
  - *Support Report*
  - *Debug*
  - *Telnet 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 Net-

work 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
- **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
- **ssl** (`cterasdk.edge.ssl.SSL`) – Object holding the Gateway SSL 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
- **nntp** (`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.Fireware`) – 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 str** The name of the logged in 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
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'])
```

(continues on next page)

(continued from previous page)

```
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('/stater/device', 'reboot') # doesn't require a param

"""TCP Connect"""
param = Object()
param.address = 'chopin.ctera.com'
param.port = 995 # CTPP
bgTask = filer.execute('/status/network', 'tcpconnect', param)
print(bgTask)
```

**See also:**

Execute the file-eviction process: Gateway.**force\_eviction**(), Reboot the Gateway: Gateway.**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'
```

(continues on next page)



(continued from previous page)

```

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 Gateway configuration

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

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

## 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)`

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`

```
"""
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

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

(continues on next page)

(continued from previous page)

```

"""
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.shared.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.shared.add('FTP', 'cloud/users/Service Account/FTP', acl = [everyone], export_
↪to_ftp = True)

```

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)

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*)

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

```

""" 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', 'Kryptonitel!') # without a full name, email or custom uid
filer.users.add('alice', 'W!z4rd0fOz!', 'Alice Wonderland') # including a full name
```

(continues on next page)

(continued from previous page)

```
filer.users.add('Bruce', 'GothamCity1!', 'Bruce Wayne', 'bruce.wayne@we.com', uid = ↵
↵1940) # all
```

`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', 'L3tsG3trR34dyT0Rumb13!')
```

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

"""Add Bruce Wayne and Domain Admins to the local Administrators group"""
filer.groups.add_members('Administrators', [('DU', 'bruce.wayne@we.com'), ('DG',
↵'WE\Domain Admins')])
```

`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')])
```

(continues on next page)

(continued from previous page)

```

"""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*)

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

```

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** (*str*) – The minimum id to use for mapping
- **end** (*str*) – The maximum id to use for mapping

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

**Note:** to retrieve a list of domain flat names, use Gateway.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)
```

## 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
- **cta\_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
- **cta\_license** (`cterasdk.edge.enum.License`, *optional*) – CTERA License, defaults to `cterasdk.edge.enum.License.EV16`

This method's behavior is identical to `Gateway.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  
:param str *ctera\_license*

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

---

**Note:** you can specify a license upon connecting the Gateway to CTERA Portal. See `Gateway.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 Sync

Sync.**suspend** ()

Suspend Cloud Sync

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

Sync.**unsuspend**()  
Unsuspend Cloud Sync

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

Sync.**refresh**()  
Refresh Cloud Folders

```
filer.sync.refresh()
```

### 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

:param str primary\_dns\_server, The primary DNS server :param str, optional secondary\_dns\_server, 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 Diagnostics

`Network.tcp_connect(address, port)`  
Test a TCP connection between the gateway and the provided address

#### Parameters

- **address** (*str*) – The address to test the connection to
- **port** (*int*) – The port of the address to test the connection to

```
filer.network.tcp_connect('chopin.ctera.com', 995) # CTP
filer.network.tcp_connect('dc.ctera.com', 389) # LDAP
```

### 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

`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`

```
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 keep 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()
```

`SSL.upload_cert (certificate, private_key, reboot=True, wait_for_reboot=False)`

Upload a server certificate

#### Parameters

- **certificate** (*str*) – A path to the PEM-encoded server certificate file
- **private\_key** (*str*) – A path to the PEM-encoded private key

```
"""
certificate = '/home/alice/certs/certificate.crt'
private_key = '/home/alice/certs/private.key'
"""

filer.ssl.upload_cert(certificate, private_key)
```



## 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()
```

## 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()
```

## 2.2.2 File Browser

**Table of Contents**

- *File Browser*
  - *Obtaining Access to the Gateway’s File System*
    - \* *List*
    - \* *Download*
    - \* *Create Directory*
    - \* *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’s path on the gateway
- **destination** (*str, optional*) – File destination, if it is a directory, the original file-name 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)
```

(continues on next page)

## 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

`__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 cterask 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 = 'Passw0rd!'
print(toxmlstr(user))
print(toxmlstr(user, True))
```



## 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
    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)
    post (baseurl, path, data)
```

```
put (baseurl, path, data)  
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.  
  
    base_api_url  
  
    base_file_url  
  
    db (path, name, param, use_file_url=False)  
  
    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)  
  
    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\_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 str** The name of the logged in 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`

**delete** (*url, headers=None*)

**get** (*url, params=None, headers=None, stream=None*)

**mkcol** (*url, headers=None*)

**post** (*url, headers=None, data="", urlencode=False*)

**put** (*url, headers=None, data=""*)

**upload** (*url, form\_data*)

**exception** `cterasdk.client.http.HTTPException` (*http\_error*)

Bases: Exception

**class** `cterasdk.client.http.HTTPResponse` (*response*)

Bases: object

**getcode** ()

```
    geturl()
    read()
class cteraskd.client.http.HttpClientBase (session_id_key)
    Bases: object
    dispatch (ctera_request)
    get_session_id()
    on_ssl_error (request)
    static on_timeout (attempt)
    set_session_id (session_id)
    should_trust (host, port)
    trust (_host, _port)
class cteraskd.client.http.HttpClientRequest (method, url, **kwargs)
    Bases: object
class cteraskd.client.http.HttpClientRequestDelete (url, headers=None)
    Bases: cteraskd.client.http.HttpClientRequest
class cteraskd.client.http.HttpClientRequestGet (url, params=None, headers=None,
    stream=None)
    Bases: cteraskd.client.http.HttpClientRequest
class cteraskd.client.http.HttpClientRequestMkcol (url, headers=None)
    Bases: cteraskd.client.http.HttpClientRequest
class cteraskd.client.http.HttpClientRequestPost (url, headers=None, data=None)
    Bases: cteraskd.client.http.HttpClientRequest
class cteraskd.client.http.HttpClientRequestPut (url, headers=None, data=None)
    Bases: cteraskd.client.http.HttpClientRequest
cteraskd.client.http.geturi (baseurl, path)
```

### cteraskd.client.ssl module

```
class cteraskd.client.ssl.CertificateServices
    Bases: object
    static add_trusted_cert (host, port)
    static save_cert_from_server (host, port)
```

## 3.1.2 cteraskd.common package

### 3.1.2.1 Submodules

#### cteraskd.common.datetime\_utils module

```
class cteraskd.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.Object`

Bases: `object`

## 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'
    ID = 'id'
    LIST = 'list'
    OBJ = 'obj'
    UUID = 'uuid'
    VAL = 'val'
```

## 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.FileBrowser (portal, base_path)
    Bases: cterasdk.core.base_command.BaseCommand
    Portal 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]`

**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*)

**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

**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

**get\_share\_info** (*path*)

Get share settings and recipients

**ls** (*path*)

Execute ls on the provided path

**Parameters** **path** (*str*) – Path to execute ls on

**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

**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*)

**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]`

**undelelete** (*path*)

Restore a previously deleted file or directory

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

**undelelete\_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

**walk** (*path*)

Perform walk on the provided path

**Parameters** **path** (*str*) – Path to perform walk on

### 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)`

### **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.base\_command module

```
class cterasdk.core.base_command.BaseCommand (portal)
    Bases: object

    Base class for all Portal API classes

    session ()
```

### 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)
        Find a Cloud Drive Folder

        Parameters

        • name (str) – Name of the Cloud Drive Folder to find

        • owner (str) – User name of the owner of the directory

        • include (list [str]) – List of metadata fields to include in the response

    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, deleted=False, user=None)
        List cloud drive folders

        Parameters

        • include (str, optional) – List of fields to retrieve, defaults to ['name', 'group',
          'owner']

        • deleted (str, optional) – Retrieve 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, winacIs=True*)

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
- **winacIs** (*bool, optional*) – Use Windows ACLs, defaults to True

**mkfg** (*name, user=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)

**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, include=None*)

Get a Device by its name

**Parameters**

- **device\_name** (*str*) – Name of the device to retrieve
- **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

**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

### cterasdk.core.enum module

**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.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.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.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.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.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.**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'
```

### 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

```
get(topic='system', minSeverity='info', originType='Portal', before=None, after=None)  
    Get logs from the Portal
```

#### Parameters

- **topic** (*cterasdk.core.enum.LogTopic*, *optional*) – Log topic to get, defaults to *cterasdk.core.enum.LogTopic.System*
- **minSeverity** (*cterasdk.core.enum.Severity*, *optional*) – Minimum severity of logs to get, defaults to *cterasdk.core.enum.Severity.INFO*
- **originType** (*cterasdk.core.enum.OriginType*, *optional*) – Origin type of the logs to get, defaults to *cterasdk.core.enum.OriginType.Portal*
- **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*

### 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

**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

**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)
    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)
    setValue (value)

class cterasdk.core.query.FilterType
    Bases: object

    Boolean = 'BooleanFilter'
    DateTime = 'DateTimeFilter'
    Integer = 'IntFilter'
    String = 'StringFilter'
    static fromValue (value)

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.Plans` (*portal*)  
Bases: `cterasdk.core.base_command.BaseCommand`

Global Admin Plan APIs

**default** = ['name']

**get** (*name, include=None*)  
Get a subscription plan

**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

### 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']

**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']

### cterasdk.core.session module

**class** `cterasdk.core.session.Session` (*host, context*)  
Bases: `cterasdk.lib.session_base.SessionBase`

**global\_admin** ()

**update\_tenant** (*current\_tenant*)

### cterasdk.core.types module

**class** `cterasdk.core.types.CloudFSFolderFindingHelper`  
Bases: tuple

Tuple holding the name and owner couple to search for folders

**name**  
The name of the CloudFS folder

**owner**  
The name of the owner of the CloudFS folder

**class** `cterasdk.core.types.GroupAccount` (*name, directory=None*)  
Bases: `cterasdk.core.types.PortalAccount`

**account\_type**  
The Portal Account Type

**Return** `cterasdk.core.enum.PortalAccountType` The Portal Account Type

**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

**account\_type**  
The Portal Account Type

**Return** `cterasdk.core.enum.PortalAccountType` The Portal Account Type

**static** `from_collaborator` (*collaborator*)

**is\_local**  
Is the account local

**Return** `bool` True if the account if local, otherwise False

**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.UserAccount` (*name, directory=None*)  
Bases: `cterasdk.core.types.PortalAccount`

**account\_type**  
The Portal Account Type

**Return** `cterasdk.core.enum.PortalAccountType` The Portal Account Type

### cterasdk.core.union module

`cterasdk.core.union.union` (*include, default*)

### 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*)  
Add a portal user

#### 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`

**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`

### 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

**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's path on the gateway

- **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*)

**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.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.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*)

Add a new array

### Parameters

- **array\_name** (*str*) – Name for the new array
- **level** (*RAIDLevel*) – RAID level
- **members** (*list (str)*) – Members of the array

**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 :param str,optional name: 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 keep 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

**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 Gateway 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

**is\_wizard\_enabled** ()  
Get the current configuration of the first time wizard

**Return** **bool** True if the first time wizard is enabled, else False



**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** (*str*) – The minimum id to use for mapping
- **end** (*str*) – The maximum id to use for mapping

**connect** (*domain, username, password, ou=None*)

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

**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**

### 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 cterask.edge.enum.CIFSPacketSigning
```

```
    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 cterask.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 = 'disabled'
```

```
    Documents = 'documents'
```

```
    Manual = 'manual'
```

```
class cterask.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'

**RAID\_0** = '0'

**RAID\_1** = '1'

**RAID\_5** = '5'

**RAID\_6** = '6'

**class** cterasdk.edge.enum.**ServicesConnectionState**

Bases: object

Gateway connection status

#### Variables

- **Disconnected** (*str*) – The Gateway is disconnected from CTERA Portal
- **Connected** (*str*) – The Gateway is connected to CTERA Portal

**Connected** = 'Connected'

**Disconnected** = 'Disconnected'

**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.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`

Gateway License configuration APIs

**apply** (*ctera\_license*)

Apply a license

:param str ctera\_license

**get** ()

Get the current Gateway License

**static infer** (*ctera\_license*)

### cterasdk.edge.login module

**class** `cterasdk.edge.login.Login` (*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

**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`

**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

**enable\_dhcp** ()

Enable DHCP

**get\_status** ()

Retrieve the network interface status

**ifconfig** ()

Retrieve the ip configuration

**ipconfig** ()

Retrieve the ip configuration

**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

:param str primary\_dns\_server, The primary DNS server :param str,optional secondary\_dns\_server, The secondary DNS server, defaults to None

**tcp\_connect** (*address, port*)

Test a TCP connection between the gateway and the provided address

**Parameters**

- **address** (*str*) – The address to test the connection to
- **port** (*int*) – The port of the address to test the connection to

**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*)

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

### cterasdk.edge.ssl module

**class** `cterasdk.edge.ssl.SSL` (*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway SSL APIs

**disable\_http** ()

Disable HTTP access

**enable\_http** ()

Enable HTTP access

**is\_http\_disabled** ()

Check if HTTP access is disabled

**is\_http\_enabled** ()

Check if HTTP access is enabled

**upload\_cert** (*certificate, private\_key, reboot=True, wait\_for\_reboot=False*)

Upload a server certificate

**Parameters**

- **certificate** (*str*) – A path to the PEM-encoded server certificate file
- **private\_key** (*str*) – A path to the PEM-encoded private key

### 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 cterasdsk.edge.query.QueryParam
    Bases: cterasdk.common.object.Object

    include_classname ()

    increment ()
```

```
class cterasdsk.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 cterasdsk.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 cterasdsk.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)
    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`

**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

**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

**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*)

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

**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

**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\_share\_winacls** (*name*)

Set a network share to use Windows ACL Emulation Mode

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

### cterasdk.edge.shell module

**class** `cterasdk.edge.shell.Shell` (*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Shell command

**run\_command** (*shell\_command*)

Execute a shell command on the gateway

**Parameters** **shell\_command** (*str*) – The shell command to execute

**Returns** The command result

### cterasdk.edge.smb module

**class** `cterasdk.edge.smb.SMB` (*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway SMB configuration APIs

**disable** ()

Disble 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*)  
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

**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.Sync (gateway)  
    Bases: cterasdk.edge.base_command.BaseCommand  
  
    Gateway Cloud Sync APIs  
  
    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  
  
    suspend ()  
        Suspend Cloud Sync
```

**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: `object`

**increment()**

**static resolve** (*task*)

**static tid** (*ref*)

**wait()**

**exception** `cterasdk.edge.taskmgr.TaskError` (*task*)  
Bases: `cterasdk.exception.CTERAException`

**class** `cterasdk.edge.taskmgr.TaskStatusEnum`

Bases: `object`

**Completed** = `'completed'`

**Failed** = `'failed'`

**Running** = `'running'`

`cterasdk.edge.taskmgr.by_name` (*CTERAHost, name*)

`cterasdk.edge.taskmgr.running` (*CTERAHost*)

`cterasdk.edge.taskmgr.wait` (*CTERAHost, ref*)

### `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.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*)

**name**

**perm**

**principal\_type**

**to\_server\_object** ()

**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*)

**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 :param str,optional name: 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*)

**static** `exists` (*filepath*)

**static** `expanduser` (*path*)

`get_dirpath` ()

**static** `get_local_file_info` (*local\_file*)

**static** `instance` ()

`rename` (*dirpath, src, dst*)

```
save (dirpath, filename, handle)  
static split_file_directory (path)  
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

    authenticated()
    initializing()
    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, sec-
                             onds=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  
  
    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
- **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
- **ssl** (`cterasdk.edge.ssl.SSL`) – Object holding the Gateway SSL 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.Fireware`) – Object holding the Gateway Firmware APIs

**base\_api\_url**

**base\_file\_url**

**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** cterasdsk.object.Portal.**GlobalAdmin** (*host, port=443, 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

**context****file\_browser\_base\_path****class** cterasdsk.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
- **files** (*cterasdk.core.files.browser.FileBrowser*) – Object holding the Portal File Browsing APIs

**base\_api\_url****base\_file\_url****base\_portal\_url****context****file\_browser\_base\_path****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=443, https=True*)  
Bases: `cterasdk.object.Portal.Portal`

Main class for Service operations on a Portal

**context**

**file\_browser\_base\_path**

## 3.1.8 cterasdk.transcript package

### 3.1.8.1 Submodules

#### cterasdk.transcript.transcribe module

`cterasdk.transcript.transcribe.prettyify` (*data*)

`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.ParserException` (*fmt, payload*)  
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.SSLErrorException` (*host, port, reason*)  
Bases: `cterasdk.exception.CTERAConnectionError`

## CHAPTER 4

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## CHAPTER 5

---

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!



### C

- cterasdk, 51
- cterasdk.client, 51
  - cterasdk.client.cteraclient, 51
  - cterasdk.client.host, 52
  - cterasdk.client.http, 53
  - cterasdk.client.ssl, 54
- cterasdk.common, 54
  - cterasdk.common.datetime\_utils, 54
  - cterasdk.common.item, 55
  - cterasdk.common.object, 55
  - cterasdk.config, 117
  - cterasdk.convert, 55
    - cterasdk.convert.exception, 55
    - cterasdk.convert.format, 55
    - cterasdk.convert.parse, 56
    - cterasdk.convert.xml\_types, 56
  - cterasdk.core, 56
    - cterasdk.core.activation, 61
    - cterasdk.core.base\_command, 62
    - cterasdk.core.cloudfs, 62
    - cterasdk.core.connection, 63
    - cterasdk.core.decorator, 63
    - cterasdk.core.devices, 63
    - cterasdk.core.directoryservice, 65
    - cterasdk.core.enum, 65
    - cterasdk.core.files, 56
      - cterasdk.core.files.browser, 56
      - cterasdk.core.files.collaboration, 59
      - cterasdk.core.files.common, 59
      - cterasdk.core.files.cp, 60
      - cterasdk.core.files.directory, 60
      - cterasdk.core.files.file\_access, 60
      - cterasdk.core.files.ln, 60
      - cterasdk.core.files.ls, 60
      - cterasdk.core.files.mv, 60
      - cterasdk.core.files.path, 61
      - cterasdk.core.files.recover, 61
      - cterasdk.core.files.rename, 61
      - cterasdk.core.files.rm, 61
    - cterasdk.core.login, 70
    - cterasdk.core.logs, 70
    - cterasdk.core.plans, 74
    - cterasdk.core.portals, 70
    - cterasdk.core.query, 72
    - cterasdk.core.remote, 74
    - cterasdk.core.reports, 73
    - cterasdk.core.servers, 74
    - cterasdk.core.session, 74
    - cterasdk.core.types, 74
    - cterasdk.core.union, 76
    - cterasdk.core.users, 76
    - cterasdk.core.zones, 77
  - cterasdk.edge, 78
    - cterasdk.edge.afp, 80
    - cterasdk.edge.aio, 80
    - cterasdk.edge.array, 81
    - cterasdk.edge.audit, 81
    - cterasdk.edge.backup, 82
    - cterasdk.edge.base\_command, 83
    - cterasdk.edge.cache, 83
    - cterasdk.edge.cli, 84
    - cterasdk.edge.config, 84
    - cterasdk.edge.connection, 85
    - cterasdk.edge.decorator, 85
    - cterasdk.edge.directoryservice, 85
    - cterasdk.edge.drive, 86
    - cterasdk.edge.enum, 86
    - cterasdk.edge.files, 78
      - cterasdk.edge.files.browser, 78
      - cterasdk.edge.files.file\_access, 79
      - cterasdk.edge.files.mkdir, 79
      - cterasdk.edge.files.path, 80
      - cterasdk.edge.files.rm, 80
    - cterasdk.edge.firmware, 95
    - cterasdk.edge.ftp, 94
    - cterasdk.edge.groups, 95
    - cterasdk.edge.licenses, 96
    - cterasdk.edge.login, 96

- cterasdk.edge.logs, 96
- cterasdk.edge.mail, 97
- cterasdk.edge.network, 97
- cterasdk.edge.nfs, 98
- cterasdk.edge.ntp, 98
- cterasdk.edge.power, 99
- cterasdk.edge.query, 100
- cterasdk.edge.remote, 100
- cterasdk.edge.rsync, 100
- cterasdk.edge.services, 101
- cterasdk.edge.session, 102
- cterasdk.edge.shares, 102
- cterasdk.edge.shell, 104
- cterasdk.edge.smb, 104
- cterasdk.edge.ssl, 99
- cterasdk.edge.support, 105
- cterasdk.edge.sync, 106
- cterasdk.edge.syslog, 107
- cterasdk.edge.taskmgr, 107
- cterasdk.edge.telnet, 108
- cterasdk.edge.timezone, 108
- cterasdk.edge.types, 108
- cterasdk.edge.uri, 109
- cterasdk.edge.users, 109
- cterasdk.edge.volumes, 110
- cterasdk.exception, 117
- cterasdk.lib, 111
  - cmd, 111
  - consent, 111
  - file\_access\_base, 112
  - filesystem, 111
  - iterator, 112
  - platform, 112
  - registry, 112
  - session\_base, 113
  - tempfile, 113
  - tracker, 113
  - version, 114
- cterasdk.object, 114
  - Agent, 114
  - Gateway, 114
  - Portal, 116
- cterasdk.transcript, 117
  - transcribe, 117



## A

- aapi (*cterasdk.edge.support.DebugLevel* attribute), 105
- Access (*cterasdk.core.enum.LogTopic* attribute), 67
- account\_type (*cterasdk.core.types.GroupAccount* attribute), 75
- account\_type (*cterasdk.core.types.PortalAccount* attribute), 75
- account\_type (*cterasdk.core.types.UserAccount* attribute), 76
- Acl (*class in cteraskd.edge.enum*), 86
- ActionResourcesParam (*class in cteraskd.core.files.common*), 59
- activate() (*cterasdk.edge.services.Services* method), 101
- Activation (*class in cteraskd.core.activation*), 61
- Active (*cterasdk.lib.session\_base.SessionStatus* attribute), 113
- add() (*cterasdk.client.host.CTERAHost* method), 52
- add() (*cterasdk.core.files.common.ActionResourcesParam* method), 59
- add() (*cterasdk.core.portals.Portals* method), 70
- add() (*cterasdk.core.users.Users* method), 76
- add() (*cterasdk.core.zones.Zones* method), 77
- add() (*cterasdk.edge.array.Array* method), 81
- add() (*cterasdk.edge.shares.Shares* method), 102
- add() (*cterasdk.edge.users.Users* method), 109
- add() (*cterasdk.edge.volumes.Volumes* method), 110
- add\_acl() (*cterasdk.edge.shares.Shares* method), 103
- add\_devices() (*cterasdk.core.zones.Zones* method), 78
- add\_first\_user() (*cterasdk.edge.users.Users* method), 110
- add\_folders() (*cterasdk.core.zones.Zones* method), 78
- add\_members() (*cterasdk.edge.groups.Groups* method), 95
- add\_share\_recipients() (*cterasdk.core.files.browser.FileBrowser* method), 56
- add\_share\_recipients() (*in module cteraskd.core.files.collaboration*), 59
- add\_trusted\_cert() (*cterasdk.client.ssl.CertificateServices* static method), 54
- addFilter() (*cterasdk.core.query.QueryParamBuilder* method), 72
- admin (*cterasdk.core.enum.Context* attribute), 66
- Administrators (*cterasdk.edge.enum.LocalGroup* attribute), 89
- advanced\_mapping() (*cterasdk.edge.directoryservice.DirectoryService* method), 85
- AFP (*class in cteraskd.edge.afp*), 80
- after() (*cterasdk.core.query.FilterBuilder* method), 72
- Agent (*class in cteraskd.object.Agent*), 114
- Agents (*cterasdk.core.enum.DeviceType* attribute), 66
- agents() (*cterasdk.core.devices.Devices* method), 63
- AIO (*class in cteraskd.edge.aio*), 80
- ALERT (*cterasdk.core.enum.Severity* attribute), 69
- ALERT (*cterasdk.edge.enum.Severity* attribute), 91
- alert (*cterasdk.edge.support.DebugLevel* attribute), 105
- ALL (*cterasdk.core.enum.PolicyType* attribute), 68
- allPortals() (*cterasdk.core.query.QueryParamBuilder* method), 72
- api() (*in module cteraskd.edge.uri*), 109
- apply() (*cterasdk.edge.licenses.Licenses* method), 96
- apps (*cterasdk.edge.support.DebugLevel* attribute), 105
- arch() (*cterasdk.lib.platform.Platform* method), 112
- Array (*class in cteraskd.edge.array*), 81
- as\_header() (*cterasdk.lib.version.Version* method), 114
- ask() (*in module cteraskd.lib.consent*), 111
- ATT (*cterasdk.convert.xml\_types.XMLTypes* attribute), 56
- Attached (*cterasdk.edge.enum.BackupConfStatusID* attribute), 87
- AttachEncrypted, 82
- Attaching (*cterasdk.edge.enum.BackupConfStatusID*

attribute), 88  
 AttachRC (class in *cterasdk.edge.backup*), 82  
 Audit (class in *cterasdk.edge.audit*), 81  
 Audit (*cterasdk.core.enum.LogTopic* attribute), 67  
 AuditEvents (class in *cterasdk.edge.enum*), 86  
 auth (*cterasdk.edge.support.DebugLevel* attribute), 105  
 authenticated() (*cterasdk.lib.session\_base.SessionBase* method), 113  
 authenticated() (in module *cterasdk.client.host*), 53  
 authenticated() (in module *cterasdk.edge.decorator*), 85  
 av (*cterasdk.edge.support.DebugLevel* attribute), 105

## B

Backup (class in *cterasdk.edge.backup*), 82  
 backup (*cterasdk.edge.support.DebugLevel* attribute), 105  
 BackupConfStatusID (class in *cterasdk.edge.enum*), 87  
 base\_api\_url (*cterasdk.client.host.CTERAHost* attribute), 52  
 base\_api\_url (*cterasdk.object.Agent.Agent* attribute), 114  
 base\_api\_url (*cterasdk.object.Gateway.Gateway* attribute), 115  
 base\_api\_url (*cterasdk.object.Portal.Portal* attribute), 116  
 base\_file\_url (*cterasdk.client.host.CTERAHost* attribute), 52  
 base\_file\_url (*cterasdk.object.Gateway.Gateway* attribute), 115  
 base\_file\_url (*cterasdk.object.Portal.Portal* attribute), 116  
 base\_portal\_url (*cterasdk.object.Portal.Portal* attribute), 116  
 BaseCommand (class in *cterasdk.core.base\_command*), 62  
 BaseCommand (class in *cterasdk.edge.base\_command*), 83  
 baseurl() (*cterasdk.client.host.NetworkHost* method), 53  
 before() (*cterasdk.core.query.FilterBuilder* method), 72  
 block\_files() (*cterasdk.edge.shares.Shares* method), 103  
 Boolean (*cterasdk.core.query.FilterType* attribute), 72  
 Boot (class in *cterasdk.edge.power*), 99  
 browse() (*cterasdk.core.portals.Portals* method), 71  
 browse\_global\_admin() (*cterasdk.core.portals.Portals* method), 71  
 build() (*cterasdk.core.query.QueryParamBuilder* method), 72

build() (*cterasdk.edge.query.QueryParamBuilder* method), 100  
 by\_name() (*cterasdk.core.devices.Devices* method), 64  
 by\_name() (in module *cterasdk.edge.taskmgr*), 108

## C

C200 (*cterasdk.core.enum.DeviceType* attribute), 66  
 C400 (*cterasdk.core.enum.DeviceType* attribute), 66  
 C800 (*cterasdk.core.enum.DeviceType* attribute), 66  
 C800P (*cterasdk.core.enum.DeviceType* attribute), 66  
 Cache (class in *cterasdk.edge.cache*), 83  
 caching (*cterasdk.edge.support.DebugLevel* attribute), 105  
 CachingGateway (*cterasdk.edge.enum.OperationMode* attribute), 90  
 CatalogReadOnlyMode (*cterasdk.edge.enum.SyncStatus* attribute), 92  
 cbck (*cterasdk.edge.support.DebugLevel* attribute), 105  
 CertificateServices (class in *cterasdk.client.ssl*), 54  
 ChangeOwner (*cterasdk.edge.enum.AuditEvents* attribute), 87  
 ChangePermissions (*cterasdk.edge.enum.AuditEvents* attribute), 87  
 CheckCodeInCorrect (*cterasdk.edge.backup.AttachRC* attribute), 82  
 Checking (*cterasdk.edge.enum.VolumeStatus* attribute), 94  
 CheckingQuota (*cterasdk.edge.enum.VolumeStatus* attribute), 94  
 CIFSPacketSigning (class in *cterasdk.edge.enum*), 88  
 CLASS (*cterasdk.convert.xml\_types.XMLTypes* attribute), 56  
 CLI (class in *cterasdk.edge.cli*), 84  
 ClientSideCaching (class in *cterasdk.edge.enum*), 88  
 ClocksOutOfSync, 82  
 ClocksOutOfSync (*cterasdk.edge.backup.AttachRC* attribute), 82  
 ClocksOutOfSync (*cterasdk.edge.enum.BackupConfStatusID* attribute), 88  
 ClocksOutOfSync (*cterasdk.edge.enum.SyncStatus* attribute), 92  
 cloud\_extender (*cterasdk.edge.support.DebugLevel* attribute), 105  
 CloudBackup (*cterasdk.core.enum.LogTopic* attribute), 67  
 CloudFS (class in *cterasdk.core.cloudfs*), 62  
 CloudFSFolderFindingHelper (class in *cterasdk.core.types*), 74

- CloudPlug (*cterasdk.core.enum.DeviceType* attribute), 66
- CloudSync (*cterasdk.core.enum.LogTopic* attribute), 67
- collaboration (*cterasdk.edge.support.DebugLevel* attribute), 105
- CollaboratorType (*class in cterasdk.core.enum*), 65
- Command (*class in cterasdk.lib.cmd*), 111
- COMPLETE (*cterasdk.edge.firmware.UploadTaskStatus* attribute), 95
- Completed (*cterasdk.edge.enum.TaskStatus* attribute), 93
- Completed (*cterasdk.edge.taskmgr.TaskStatusEnum* attribute), 108
- compute\_zip\_file\_name () (*cterasdk.lib.filesystem.FileSystem* static method), 111
- Config (*class in cterasdk.edge.config*), 84
- configure () (*cterasdk.edge.backup.Backup* method), 82
- Configuring (*cterasdk.edge.enum.BackupConfStatusID* attribute), 88
- connect () (*cterasdk.edge.directoryservice.DirectoryService* method), 85
- connect () (*cterasdk.edge.services.Services* method), 101
- Connected (*cterasdk.edge.enum.ServicesConnectionState* attribute), 91
- Connected (*cterasdk.edge.enum.SyncStatus* attribute), 92
- connected () (*cterasdk.edge.services.Services* method), 101
- ConnectingFolders (*cterasdk.edge.enum.SyncStatus* attribute), 92
- ConnectionFailed (*cterasdk.edge.enum.SyncStatus* attribute), 92
- ConnectionTimeout, 118
- ConsentException, 118
- ContainsErrors (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- ContentType (*class in cterasdk.client.http*), 53
- Context (*class in cterasdk.core.enum*), 66
- context (*cterasdk.object.Portal.GlobalAdmin* attribute), 116
- context (*cterasdk.object.Portal.Portal* attribute), 116
- context (*cterasdk.object.Portal.ServicesPortal* attribute), 117
- Converting (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- copy () (*cterasdk.core.files.browser.FileBrowser* method), 57
- copy () (*in module cterasdk.core.files.cp*), 60
- copy\_multi () (*cterasdk.core.files.browser.FileBrowser* method), 57
- copy\_multi () (*in module cterasdk.core.files.cp*), 60
- Corrupted (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- countLimit () (*cterasdk.core.query.QueryParamBuilder* method), 72
- countLimit () (*cterasdk.edge.query.QueryParamBuilder* method), 100
- CreateElement () (*in module cterasdk.convert.format*), 55
- CreateFilesWriteData (*cterasdk.edge.enum.AuditEvents* attribute), 87
- CreateFolderRC (*class in cterasdk.edge.backup*), 82
- CreateFoldersAppendData (*cterasdk.edge.enum.AuditEvents* attribute), 87
- CreateShareParam (*class in cterasdk.core.files.common*), 59
- CRITICAL (*cterasdk.core.enum.Severity* attribute), 69
- CRITICAL (*cterasdk.edge.enum.Severity* attribute), 91
- CTERAClient (*class in cterasdk.client.cteraclient*), 51
- CTERAClientException, 117
- CTERAConnectionError, 117
- CTERAException, 117
- CTERAHost (*class in cterasdk.client.host*), 52
- CTERAPath (*class in cterasdk.core.files.path*), 61
- CTERAPath (*class in cterasdk.edge.files.path*), 80
- cterasdk (*module*), 51
- cterasdk.client (*module*), 51
- cterasdk.client.cteraclient (*module*), 51
- cterasdk.client.host (*module*), 52
- cterasdk.client.http (*module*), 53
- cterasdk.client.ssl (*module*), 54
- cterasdk.common (*module*), 54
- cterasdk.common.datetime\_utils (*module*), 54
- cterasdk.common.item (*module*), 55
- cterasdk.common.object (*module*), 55
- cterasdk.config (*module*), 117
- cterasdk.convert (*module*), 55
- cterasdk.convert.exception (*module*), 55
- cterasdk.convert.format (*module*), 55
- cterasdk.convert.parse (*module*), 56
- cterasdk.convert.xml\_types (*module*), 56
- cterasdk.core (*module*), 56
- cterasdk.core.activation (*module*), 61
- cterasdk.core.base\_command (*module*), 62
- cterasdk.core.cloudfs (*module*), 62
- cterasdk.core.connection (*module*), 63
- cterasdk.core.decorator (*module*), 63
- cterasdk.core.devices (*module*), 63
- cterasdk.core.directoryservice (*module*), 65
- cterasdk.core.enum (*module*), 65
- cterasdk.core.files (*module*), 56

- `cterasdk.core.files.browser (module)`, 56
  - `cterasdk.core.files.collaboration (module)`, 59
  - `cterasdk.core.files.common (module)`, 59
  - `cterasdk.core.files.cp (module)`, 60
  - `cterasdk.core.files.directory (module)`, 60
  - `cterasdk.core.files.file_access (module)`, 60
  - `cterasdk.core.files.ln (module)`, 60
  - `cterasdk.core.files.ls (module)`, 60
  - `cterasdk.core.files.mv (module)`, 60
  - `cterasdk.core.files.path (module)`, 61
  - `cterasdk.core.files.recover (module)`, 61
  - `cterasdk.core.files.rename (module)`, 61
  - `cterasdk.core.files.rm (module)`, 61
  - `cterasdk.core.login (module)`, 70
  - `cterasdk.core.logs (module)`, 70
  - `cterasdk.core.plans (module)`, 74
  - `cterasdk.core.portals (module)`, 70
  - `cterasdk.core.query (module)`, 72
  - `cterasdk.core.remote (module)`, 74
  - `cterasdk.core.reports (module)`, 73
  - `cterasdk.core.servers (module)`, 74
  - `cterasdk.core.session (module)`, 74
  - `cterasdk.core.types (module)`, 74
  - `cterasdk.core.union (module)`, 76
  - `cterasdk.core.users (module)`, 76
  - `cterasdk.core.zones (module)`, 77
  - `cterasdk.edge (module)`, 78
  - `cterasdk.edge.afp (module)`, 80
  - `cterasdk.edge.aio (module)`, 80
  - `cterasdk.edge.array (module)`, 81
  - `cterasdk.edge.audit (module)`, 81
  - `cterasdk.edge.backup (module)`, 82
  - `cterasdk.edge.base_command (module)`, 83
  - `cterasdk.edge.cache (module)`, 83
  - `cterasdk.edge.cli (module)`, 84
  - `cterasdk.edge.config (module)`, 84
  - `cterasdk.edge.connection (module)`, 85
  - `cterasdk.edge.decorator (module)`, 85
  - `cterasdk.edge.directoryservice (module)`, 85
  - `cterasdk.edge.drive (module)`, 86
  - `cterasdk.edge.enum (module)`, 86
  - `cterasdk.edge.files (module)`, 78
  - `cterasdk.edge.files.browser (module)`, 78
  - `cterasdk.edge.files.file_access (module)`, 79
  - `cterasdk.edge.files.mkdir (module)`, 79
  - `cterasdk.edge.files.path (module)`, 80
  - `cterasdk.edge.files.rm (module)`, 80
  - `cterasdk.edge.firmware (module)`, 95
  - `cterasdk.edge.ftp (module)`, 94
  - `cterasdk.edge.groups (module)`, 95
  - `cterasdk.edge.licenses (module)`, 96
  - `cterasdk.edge.login (module)`, 96
  - `cterasdk.edge.logs (module)`, 96
  - `cterasdk.edge.mail (module)`, 97
  - `cterasdk.edge.network (module)`, 97
  - `cterasdk.edge.nfs (module)`, 98
  - `cterasdk.edge.ntp (module)`, 98
  - `cterasdk.edge.power (module)`, 99
  - `cterasdk.edge.query (module)`, 100
  - `cterasdk.edge.remote (module)`, 100
  - `cterasdk.edge.rsync (module)`, 100
  - `cterasdk.edge.services (module)`, 101
  - `cterasdk.edge.session (module)`, 102
  - `cterasdk.edge.shares (module)`, 102
  - `cterasdk.edge.shell (module)`, 104
  - `cterasdk.edge.smb (module)`, 104
  - `cterasdk.edge.ssl (module)`, 99
  - `cterasdk.edge.support (module)`, 105
  - `cterasdk.edge.sync (module)`, 106
  - `cterasdk.edge.syslog (module)`, 107
  - `cterasdk.edge.taskmgr (module)`, 107
  - `cterasdk.edge.telnet (module)`, 108
  - `cterasdk.edge.timezone (module)`, 108
  - `cterasdk.edge.types (module)`, 108
  - `cterasdk.edge.uri (module)`, 109
  - `cterasdk.edge.users (module)`, 109
  - `cterasdk.edge.volumes (module)`, 110
  - `cterasdk.exception (module)`, 117
  - `cterasdk.lib (module)`, 111
  - `cterasdk.lib.cmd (module)`, 111
  - `cterasdk.lib.consent (module)`, 111
  - `cterasdk.lib.file_access_base (module)`, 112
  - `cterasdk.lib.filesystem (module)`, 111
  - `cterasdk.lib.iterator (module)`, 112
  - `cterasdk.lib.platform (module)`, 112
  - `cterasdk.lib.registry (module)`, 112
  - `cterasdk.lib.session_base (module)`, 113
  - `cterasdk.lib.tempfile (module)`, 113
  - `cterasdk.lib.tracker (module)`, 113
  - `cterasdk.lib.version (module)`, 114
  - `cterasdk.object (module)`, 114
  - `cterasdk.object.Agent (module)`, 114
  - `cterasdk.object.Gateway (module)`, 114
  - `cterasdk.object.Portal (module)`, 116
  - `cterasdk.transcript (module)`, 117
  - `cterasdk.transcript.transcribe (module)`, 117
  - `cttp (cterasdk.edge.support.DebugLevel attribute)`, 105
  - `cttp_data (cterasdk.edge.support.DebugLevel attribute)`, 105
- ## D
- `Dateime (cterasdk.core.query.FilterType attribute)`, 72

- DateTimeUtils (class in *cterasdk.common.datetime\_utils*), 54
- db (*cterasdk.edge.support.DebugLevel* attribute), 105
- db () (*cterasdk.client.cteraclient.CTERAClient* method), 51
- db () (*cterasdk.client.host.CTERAHost* method), 52
- DEBUG (*cterasdk.core.enum.Severity* attribute), 69
- DEBUG (*cterasdk.edge.enum.Severity* attribute), 91
- debug (*cterasdk.edge.support.DebugLevel* attribute), 105
- DebugLevel (class in *cterasdk.edge.support*), 105
- default (*cterasdk.core.cloudfs.CloudFS* attribute), 62
- default (*cterasdk.core.devices.Devices* attribute), 64
- default (*cterasdk.core.plans.Plans* attribute), 74
- default (*cterasdk.core.portals.Portals* attribute), 71
- default (*cterasdk.core.servers.Servers* attribute), 74
- default (*cterasdk.core.users.Users* attribute), 77
- default\_include (*cterasdk.edge.logs.Logs* attribute), 96
- defaultAuditEvents (*cterasdk.edge.audit.Audit* attribute), 81
- Delete (*cterasdk.edge.enum.AuditEvents* attribute), 87
- delete () (*cterasdk.client.cteraclient.CTERAClient* method), 51
- delete () (*cterasdk.client.host.CTERAHost* method), 52
- delete () (*cterasdk.client.http.HTTPClient* method), 53
- delete () (*cterasdk.core.cloudfs.CloudFS* method), 62
- delete () (*cterasdk.core.files.browser.FileBrowser* method), 57
- delete () (*cterasdk.core.portals.Portals* method), 71
- delete () (*cterasdk.core.users.Users* method), 77
- delete () (*cterasdk.core.zones.Zones* method), 78
- delete () (*cterasdk.edge.array.Array* method), 81
- delete () (*cterasdk.edge.files.browser.FileBrowser* method), 78
- delete () (*cterasdk.edge.shares.Shares* method), 103
- delete () (*cterasdk.edge.users.Users* method), 110
- delete () (*cterasdk.edge.volumes.Volumes* method), 111
- delete () (in module *cterasdk.core.files.rm*), 61
- delete () (in module *cterasdk.edge.files.rm*), 80
- delete\_all () (*cterasdk.edge.array.Array* method), 81
- delete\_all () (*cterasdk.edge.volumes.Volumes* method), 111
- delete\_multi () (*cterasdk.core.files.browser.FileBrowser* method), 57
- delete\_multi () (in module *cterasdk.core.files.rm*), 61
- DeleteSubfoldersAndFiles (*cterasdk.edge.enum.AuditEvents* attribute), 87
- desktops () (*cterasdk.core.devices.Devices* method), 64
- Device (*cterasdk.core.enum.OriginType* attribute), 67
- device () (*cterasdk.core.devices.Devices* method), 64
- Devices (class in *cterasdk.core.devices*), 63
- devices () (*cterasdk.core.devices.Devices* method), 64
- DeviceType (class in *cterasdk.core.enum*), 66
- df () (*cterasdk.config.Logging* static method), 117
- DG (*cterasdk.core.enum.CollaboratorType* attribute), 66
- DG (*cterasdk.edge.enum.PrincipalType* attribute), 90
- DirectoryService (class in *cterasdk.core.directoryservice*), 65
- DirectoryService (class in *cterasdk.edge.directoryservice*), 85
- disable () (*cterasdk.config.Logging* static method), 117
- disable () (*cterasdk.edge.afp.AFP* method), 80
- disable () (*cterasdk.edge.aio.AIO* method), 80
- disable () (*cterasdk.edge.audit.Audit* method), 81
- disable () (*cterasdk.edge.cache.Cache* method), 83
- disable () (*cterasdk.edge.ftp.FTP* method), 94
- disable () (*cterasdk.edge.mail.Mail* method), 97
- disable () (*cterasdk.edge.nfs.NFS* method), 98
- disable () (*cterasdk.edge.ntp.NTP* method), 99
- disable () (*cterasdk.edge.rsync.RSync* method), 100
- disable () (*cterasdk.edge.smb.SMB* method), 104
- disable () (*cterasdk.edge.syslog.Syslog* method), 107
- disable () (*cterasdk.edge.telnet.Telnet* method), 108
- disable\_abe () (*cterasdk.edge.smb.SMB* method), 104
- disable\_http () (*cterasdk.edge.ssl.SSL* method), 99
- disable\_remote\_access () (*cterasdk.edge.session.Session* method), 102
- disable\_sso () (*cterasdk.edge.services.Services* method), 101
- disable\_wizard () (*cterasdk.edge.config.Config* method), 84
- Disabled (*cterasdk.core.enum.Role* attribute), 69
- Disabled (*cterasdk.edge.enum.CIFSPacketSigning* attribute), 88
- Disabled (*cterasdk.edge.enum.ClientSideCaching* attribute), 88
- Disabled (*cterasdk.edge.enum.Mode* attribute), 90
- Disabled (*cterasdk.edge.enum.OperationMode* attribute), 90
- disconnect () (*cterasdk.edge.directoryservice.DirectoryService* method), 85
- disconnect () (*cterasdk.edge.services.Services* method), 101
- Disconnected (*cterasdk.edge.enum.ServicesConnectionState* attribute), 91
- DisconnectedPortal (*cterasdk.edge.enum.SyncStatus* attribute), 92

- dispatch() (*cterasdk.client.http.HttpClientBase method*), 54  
 dns (*cterasdk.edge.support.DebugLevel attribute*), 105  
 Documents (*cterasdk.edge.enum.ClientSideCaching attribute*), 88  
 domain\_group() (*cterasdk.core.types.ShareRecipient static method*), 75  
 domain\_user() (*cterasdk.core.types.ShareRecipient static method*), 75  
 domains() (*cterasdk.edge.directoryservice.DirectoryService method*), 86  
 download() (*cterasdk.client.cteraclient.CTERAClient method*), 51  
 download() (*cterasdk.core.files.browser.FileBrowser method*), 57  
 download() (*cterasdk.edge.files.browser.FileBrowser method*), 78  
 download() (*cterasdk.lib.file\_access\_base.FileAccessBase method*), 112  
 download\_as\_zip() (*cterasdk.core.files.browser.FileBrowser method*), 57  
 download\_as\_zip() (*cterasdk.edge.files.browser.FileBrowser method*), 79  
 download\_as\_zip() (*cterasdk.lib.file\_access\_base.FileAccessBase method*), 112  
 download\_zip() (*cterasdk.client.cteraclient.CTERAClient method*), 51  
 download\_zip() (*cterasdk.client.host.CTERAHost method*), 52  
 Drive (*class in cterasdk.edge.drive*), 86  
 DU (*cterasdk.core.enum.CollaboratorType attribute*), 66  
 DU (*cterasdk.edge.enum.PrincipalType attribute*), 90
- ## E
- Email (*cterasdk.core.enum.ProtectionLevel attribute*), 68  
 EMERGENCY (*cterasdk.core.enum.Severity attribute*), 69  
 EMERGENCY (*cterasdk.edge.enum.Severity attribute*), 91  
 enable() (*cterasdk.config.Logging static method*), 117  
 enable() (*cterasdk.edge.aio.AIO method*), 80  
 enable() (*cterasdk.edge.audit.Audit method*), 81  
 enable() (*cterasdk.edge.cache.Cache method*), 83  
 enable() (*cterasdk.edge.ftp.FTP method*), 94  
 enable() (*cterasdk.edge.mail.Mail method*), 97  
 enable() (*cterasdk.edge.nfs.NFS method*), 98  
 enable() (*cterasdk.edge.ntp.NTP method*), 99  
 enable() (*cterasdk.edge.rsync.RSync method*), 100  
 enable() (*cterasdk.edge.smb.SMB method*), 104  
 enable() (*cterasdk.edge.syslog.Syslog method*), 107  
 enable() (*cterasdk.edge.telnet.Telnet method*), 108  
 enable\_abe() (*cterasdk.edge.smb.SMB method*), 104  
 enable\_dhcp() (*cterasdk.edge.network.Network method*), 97  
 enable\_http() (*cterasdk.edge.ssl.SSL method*), 99  
 enable\_remote\_access() (*cterasdk.edge.session.Session method*), 102  
 enable\_sso() (*cterasdk.edge.services.Services method*), 101  
 enable\_wizard() (*cterasdk.edge.config.Config method*), 84  
 Enabled (*cterasdk.edge.enum.Mode attribute*), 90  
 encoded\_fullpath() (*cterasdk.core.files.path.CTERAPath method*), 61  
 encoded\_fullpath() (*cterasdk.edge.files.path.CTERAPath method*), 80  
 encoded\_parent() (*cterasdk.core.files.path.CTERAPath method*), 61  
 encoded\_parent() (*cterasdk.edge.files.path.CTERAPath method*), 80  
 EncryptionMode (*class in cterasdk.edge.backup*), 83  
 EndUser (*cterasdk.core.enum.Role attribute*), 69  
 eq() (*cterasdk.core.query.FilterBuilder method*), 72  
 EQUALS (*cterasdk.core.query.Restriction attribute*), 73  
 ERROR (*cterasdk.core.enum.Severity attribute*), 69  
 ERROR (*cterasdk.edge.enum.Severity attribute*), 91  
 error (*cterasdk.edge.support.DebugLevel attribute*), 105  
 error\_abort (*cterasdk.edge.support.DebugLevel attribute*), 105  
 ErrorStatus, 113  
 EV128 (*cterasdk.edge.enum.License attribute*), 89  
 EV16 (*cterasdk.edge.enum.License attribute*), 89  
 EV32 (*cterasdk.edge.enum.License attribute*), 89  
 EV4 (*cterasdk.edge.enum.License attribute*), 89  
 EV64 (*cterasdk.edge.enum.License attribute*), 89  
 EV8 (*cterasdk.edge.enum.License attribute*), 89  
 Everyone (*cterasdk.edge.enum.LocalGroup attribute*), 89  
 evictor (*cterasdk.edge.support.DebugLevel attribute*), 106  
 evictor\_verbose (*cterasdk.edge.support.DebugLevel attribute*), 106  
 execute() (*cterasdk.client.cteraclient.CTERAClient method*), 51  
 execute() (*cterasdk.client.host.CTERAHost method*), 52  
 ExhaustedException, 118  
 exists() (*cterasdk.lib.filesystem.FileSystem static method*), 111  
 expanduser() (*cterasdk.lib.filesystem.FileSystem static method*), 111  
 expire\_in() (*cterasdk.core.types.ShareRecipient*

- method), 75
- expire\_on() (cterasdk.core.types.ShareRecipient method), 75
- export() (cterasdk.edge.config.Config method), 84
- EXT (cterasdk.core.enum.CollaboratorType attribute), 66
- external() (cterasdk.core.types.ShareRecipient static method), 75
- ## F
- FAIL (cterasdk.edge.firmware.UploadTaskStatus attribute), 95
- Failed (cterasdk.edge.enum.BackupConfStatusID attribute), 88
- Failed (cterasdk.edge.enum.TaskStatus attribute), 93
- Failed (cterasdk.edge.taskmgr.TaskStatusEnum attribute), 108
- failed() (cterasdk.lib.tracker.StatusTracker method), 113
- FailedFilesInReadOnlyFolder (cterasdk.edge.enum.SyncStatus attribute), 92
- fetch() (cterasdk.core.directoryservice.DirectoryService method), 65
- fetch\_resources() (in module cterasdk.core.files.ls), 60
- file\_browser\_base\_path (cterasdk.object.Portal.GlobalAdmin attribute), 116
- file\_browser\_base\_path (cterasdk.object.Portal.Portal attribute), 116
- file\_browser\_base\_path (cterasdk.object.Portal.ServicesPortal attribute), 117
- file\_descriptor() (cterasdk.client.cteraclient.CTERAClient static method), 51
- FileAccess (class in cterasdk.core.files.file\_access), 60
- FileAccess (class in cterasdk.edge.files.file\_access), 79
- FileAccessBase (class in cterasdk.lib.file\_access\_base), 112
- FileAccessMode (class in cterasdk.core.enum), 67
- FileAccessMode (class in cterasdk.edge.enum), 88
- FileBrowser (class in cterasdk.core.files.browser), 56
- FileBrowser (class in cterasdk.edge.files.browser), 78
- filers() (cterasdk.core.devices.Devices method), 65
- files (cterasdk.edge.support.DebugLevel attribute), 106
- files() (in module cterasdk.edge.uri), 109
- FileSystem (class in cterasdk.lib.filesystem), 111
- FileSystemException, 118
- Filter (class in cterasdk.core.query), 72
- FilterBuilder (class in cterasdk.core.query), 72
- FilterType (class in cterasdk.core.query), 72
- find() (cterasdk.core.cloudfs.CloudFS method), 62
- Firmware (class in cterasdk.edge.firmware), 95
- fmt() (cterasdk.config.Logging static method), 117
- folder\_groups() (cterasdk.core.reports.Reports method), 73
- FolderAlreadyExists (cterasdk.edge.backup.CreateFolderRC attribute), 82
- folders() (cterasdk.core.reports.Reports method), 73
- Forbidden, 79
- force\_eviction() (cterasdk.edge.cache.Cache method), 83
- form\_data() (cterasdk.client.cteraclient.CTERAClient method), 51
- form\_data() (cterasdk.client.host.CTERAHost method), 52
- format() (cterasdk.edge.drive.Drive method), 86
- format\_all() (cterasdk.edge.drive.Drive method), 86
- Formatting (cterasdk.edge.enum.VolumeStatus attribute), 94
- from\_collaborator() (cterasdk.core.types.PortalAccount static method), 75
- from\_server\_object() (cterasdk.edge.types.ShareAccessControlEntry static method), 109
- from\_server\_object() (cterasdk.edge.types.UserGroupEntry static method), 109
- fromjsonstr() (in module cterasdk.convert.parse), 56
- fromValue() (cterasdk.core.query.FilterType static method), 72
- fromxmlstr() (cterasdk.client.cteraclient.CTERAClient static method), 51
- fromxmlstr() (in module cterasdk.convert.parse), 56
- FTP (class in cterasdk.edge.ftp), 94
- fullpath() (cterasdk.core.files.path.CTERAPath method), 61
- fullpath() (cterasdk.edge.files.path.CTERAPath method), 80
- ## G
- Gateway (class in cterasdk.object.Gateway), 114
- Gateways (cterasdk.core.enum.DeviceType attribute), 66
- ge() (cterasdk.core.query.FilterBuilder method), 72
- generate\_code() (cterasdk.core.activation.Activation method), 61
- get() (cterasdk.client.cteraclient.CTERAClient method), 51
- get() (cterasdk.client.host.CTERAHost method), 52

- get () (*cterasdk.client.http.HTTPClient method*), 53  
 get () (*cterasdk.config.Logging static method*), 117  
 get () (*cterasdk.core.logs.Logs method*), 70  
 get () (*cterasdk.core.plans.Plans method*), 74  
 get () (*cterasdk.core.users.Users method*), 77  
 get () (*cterasdk.core.zones.Zones method*), 78  
 get () (*cterasdk.edge.array.Array method*), 81  
 get () (*cterasdk.edge.drive.Drive method*), 86  
 get () (*cterasdk.edge.groups.Groups method*), 96  
 get () (*cterasdk.edge.licenses.Licenses method*), 96  
 get () (*cterasdk.edge.shares.Shares method*), 103  
 get () (*cterasdk.edge.users.Users method*), 110  
 get () (*cterasdk.edge.volumes.Volumes method*), 111  
 get () (*cterasdk.lib.registry.Registry method*), 112  
 get\_configuration () (*cterasdk.edge.ftp.FTP method*), 94  
 get\_configuration () (*cterasdk.edge.nfs.NFS method*), 98  
 get\_configuration () (*cterasdk.edge.rsync.RSync method*), 100  
 get\_configuration () (*cterasdk.edge.smb.SMB method*), 105  
 get\_configuration () (*cterasdk.edge.syslog.Syslog method*), 107  
 get\_connected\_domain () (*cterasdk.edge.directoryservice.DirectoryService method*), 86  
 get\_dirpath () (*cterasdk.lib.filesystem.FileSystem method*), 111  
 get\_expiration\_date () (*cterasdk.common.datetime\_utils.DateTimeUtils static method*), 54  
 get\_hostname () (*cterasdk.edge.config.Config method*), 84  
 get\_local\_file\_info () (*cterasdk.lib.filesystem.FileSystem static method*), 111  
 get\_location () (*cterasdk.edge.config.Config method*), 84  
 get\_multi () (*cterasdk.client.cteraclient.CTERAClient method*), 51  
 get\_multi () (*cterasdk.client.host.CTERAHost method*), 52  
 get\_resource\_info () (*in module cterasdk.core.files.common*), 59  
 get\_session\_id () (*cterasdk.client.cteraclient.CTERAClient method*), 51  
 get\_session\_id () (*cterasdk.client.host.CTERAHost method*), 52  
 get\_session\_id () (*cterasdk.client.http.HttpClientBase method*), 54  
 get\_share\_info () (*cterasdk.core.files.browser.FileBrowser method*), 57  
 get\_share\_info () (*in module cterasdk.core.files.collaboration*), 59  
 get\_status () (*cterasdk.edge.drive.Drive method*), 86  
 get\_status () (*cterasdk.edge.network.Network method*), 97  
 get\_status () (*cterasdk.edge.services.Services method*), 101  
 get\_status () (*cterasdk.edge.sync.Sync method*), 106  
 get\_support\_report () (*cterasdk.edge.support.Support method*), 106  
 get\_timezone () (*cterasdk.edge.timezone.Timezone method*), 108  
 getcode () (*cterasdk.client.http.HTTPResponse method*), 53  
 GetFoldersList (*cterasdk.edge.enum.BackupConfStatusID attribute*), 88  
 geturi () (*in module cterasdk.client.http*), 54  
 geturl () (*cterasdk.client.http.HTTPResponse method*), 53  
 global\_admin () (*cterasdk.core.session.Session method*), 74  
 GlobalAdmin (*class in cterasdk.object.Portal*), 116  
 GREATER\_EQUALS (*cterasdk.core.query.Restriction attribute*), 73  
 GREATER\_THAN (*cterasdk.core.query.Restriction attribute*), 73  
 Group (*cterasdk.core.enum.PortalAccountType attribute*), 68  
 GroupAccount (*class in cterasdk.core.types*), 75  
 Groups (*class in cterasdk.edge.groups*), 95  
 Groups (*cterasdk.core.enum.SearchType attribute*), 69  
 gt () (*cterasdk.core.query.FilterBuilder method*), 72
- ## H
- host () (*cterasdk.client.host.NetworkHost method*), 53  
 HostUnreachable, 118  
 http (*cterasdk.edge.support.DebugLevel attribute*), 106  
 HTTPClient (*class in cterasdk.client.http*), 53  
 HttpClientBase (*class in cterasdk.client.http*), 54  
 HttpClientRequest (*class in cterasdk.client.http*), 54  
 HttpClientRequestDelete (*class in cterasdk.client.http*), 54  
 HttpClientRequestGet (*class in cterasdk.client.http*), 54  
 HttpClientRequestMkcol (*class in cterasdk.client.http*), 54  
 HttpClientRequestPost (*class in cterasdk.client.http*), 54  
 HttpClientRequestPut (*class in cterasdk.client.http*), 54  
 HTTPException, 53  
 HTTPResponse (*class in cterasdk.client.http*), 53  
 https () (*cterasdk.client.host.NetworkHost method*), 53



- I**
- ID (*cterasdk.convert.xml\_types.XMLTypes* attribute), 56
  - IfClientAgrees (*cterasdk.edge.enum.CIFSPacketSigning* attribute), 88
  - ifconfig() (*cterasdk.edge.network.Network* method), 97
  - IN\_PROGRESS (*cterasdk.edge.firmware.UploadTaskStatus* attribute), 95
  - Inactive (*cterasdk.lib.session\_base.SessionStatus* attribute), 113
  - include() (*cterasdk.core.query.QueryParamBuilder* method), 72
  - include() (*cterasdk.edge.query.QueryParamBuilder* method), 100
  - include\_classname() (*cterasdk.core.query.QueryParamBuilder* method), 72
  - include\_classname() (*cterasdk.core.query.QueryParams* method), 73
  - include\_classname() (*cterasdk.edge.query.QueryParam* method), 100
  - IncorrectPassphrase, 83
  - increment() (*cterasdk.core.query.QueryParams* method), 73
  - increment() (*cterasdk.edge.query.QueryParam* method), 100
  - increment() (*cterasdk.edge.taskmgr.Task* method), 107
  - increment() (*cterasdk.lib.tracker.StatusTracker* method), 113
  - index (*cterasdk.edge.support.DebugLevel* attribute), 106
  - infer() (*cterasdk.edge.licenses.Licenses* static method), 96
  - INFO (*cterasdk.core.enum.Severity* attribute), 69
  - INFO (*cterasdk.edge.enum.Severity* attribute), 91
  - info (*cterasdk.edge.support.DebugLevel* attribute), 106
  - Initializing (*cterasdk.lib.session\_base.SessionStatus* attribute), 113
  - initializing() (*cterasdk.lib.session\_base.SessionBase* method), 113
  - InitializingConnection (*cterasdk.edge.enum.SyncStatus* attribute), 92
  - InputError, 118
  - instance() (*cterasdk.core.files.common.ActionResourcesParam* static method), 59
  - instance() (*cterasdk.core.files.common.CreateShareParam* static method), 59
  - instance() (*cterasdk.core.files.common.SrcDstParam* static method), 59
  - instance() (*cterasdk.lib.filesystem.FileSystem* static method), 111
  - instance() (*cterasdk.lib.platform.Platform* static method), 112
  - instance() (*cterasdk.lib.registry.Registry* static method), 112
  - instance() (*cterasdk.lib.version.Version* static method), 114
  - Integer (*cterasdk.core.query.FilterType* attribute), 72
  - InternalError (*cterasdk.edge.enum.SyncStatus* attribute), 92
  - InternalServerError (*cterasdk.edge.backup.AttachRC* attribute), 82
  - InternalServerError (*cterasdk.edge.backup.CreateFolderRC* attribute), 82
  - InvalidAverageBlockSize (*cterasdk.edge.enum.SyncStatus* attribute), 92
  - InvalidConfiguration (*cterasdk.edge.enum.SyncStatus* attribute), 92
  - InvalidName, 60
  - InvalidPath, 60
  - ipconfig() (*cterasdk.edge.network.Network* method), 97
  - IPProtocol (class in *cterasdk.edge.enum*), 89
  - is\_configured() (*cterasdk.edge.backup.Backup* method), 82
  - is\_disabled() (*cterasdk.edge.afp.AFP* method), 80
  - is\_disabled() (*cterasdk.edge.ftp.FTP* method), 94
  - is\_disabled() (*cterasdk.edge.nfs.NFS* method), 98
  - is\_disabled() (*cterasdk.edge.rsyc.RSync* method), 100
  - is\_disabled() (*cterasdk.edge.sync.Sync* method), 106
  - is\_enabled() (*cterasdk.edge.aio.AIO* method), 80
  - is\_enabled() (*cterasdk.edge.cache.Cache* method), 83
  - is\_enabled() (*cterasdk.edge.sync.Sync* method), 106
  - is\_http\_disabled() (*cterasdk.edge.ssl.SSL* method), 99
  - is\_http\_enabled() (*cterasdk.edge.ssl.SSL* method), 99
  - is\_local (*cterasdk.core.types.PortalAccount* attribute), 75
  - is\_nosession() (in *cterasdk.edge.decorators* module), 85
  - is\_wizard\_enabled() (*cterasdk.edge.config.Config* method), 84
  - IsEncrypted (*cterasdk.edge.backup.AttachRC* attribute), 82
  - Item (class in *cterasdk.common.item*), 55
  - ItemExists, 60, 79
  - Iterator (class in *cterasdk.lib.iterator*), 112

- iterator() (*cterasdk.object.Portal.Portal* method), 116
- iterator() (in module *cterasdk.core.query*), 73
- ## J
- JBOD (*cterasdk.edge.enum.RAIDLevel* attribute), 91
- join() (*cterasdk.exception.CTERAException* method), 118
- joinpath() (*cterasdk.core.files.path.CTERAPath* method), 61
- joinpath() (*cterasdk.edge.files.path.CTERAPath* method), 80
- ## K
- KeyRequired (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- ## L
- le() (*cterasdk.core.query.FilterBuilder* method), 72
- LESS\_EQUALS (*cterasdk.core.query.Restriction* attribute), 73
- LESS\_THAN (*cterasdk.core.query.Restriction* attribute), 73
- LG (*cterasdk.core.enum.CollaboratorType* attribute), 66
- LG (*cterasdk.edge.enum.PrincipalType* attribute), 90
- License (class in *cterasdk.edge.enum*), 89
- license (*cterasdk.edge.support.DebugLevel* attribute), 106
- Licenses (class in *cterasdk.edge.licenses*), 96
- LIKE (*cterasdk.core.query.Restriction* attribute), 73
- like() (*cterasdk.core.query.FilterBuilder* method), 72
- LIST (*cterasdk.convert.xml\_types.XMLTypes* attribute), 56
- list\_dir() (in module *cterasdk.core.files.ls*), 60
- list\_domain\_users() (*cterasdk.core.users.Users* method), 77
- list\_domains() (*cterasdk.core.users.Users* method), 77
- list\_folder\_groups() (*cterasdk.core.cloudfs.CloudFS* method), 62
- list\_folders() (*cterasdk.core.cloudfs.CloudFS* method), 62
- list\_local\_users() (*cterasdk.core.users.Users* method), 77
- list\_servers() (*cterasdk.core.servers.Servers* method), 74
- list\_tenants() (*cterasdk.core.portals.Portals* method), 71
- ListFolderReadData (*cterasdk.edge.enum.AuditEvents* attribute), 87
- Local (*cterasdk.edge.session.SessionType* attribute), 102
- local() (*cterasdk.edge.session.Session* method), 102
- local() (in module *cterasdk.edge.uri*), 109
- local\_group() (*cterasdk.core.types.ShareRecipient* static method), 76
- local\_user() (*cterasdk.core.types.ShareRecipient* static method), 76
- LocalDirectoryNotFound, 118
- LocalFileNotFound, 118
- LocalGroup (class in *cterasdk.edge.enum*), 89
- LocalPathNotFound, 118
- Logging (class in *cterasdk.config*), 117
- Login (class in *cterasdk.core.login*), 70
- Login (class in *cterasdk.edge.login*), 96
- login() (*cterasdk.client.host.CTERAHost* method), 52
- login() (*cterasdk.core.login.Login* method), 70
- login() (*cterasdk.edge.login.Login* method), 96
- login() (in module *cterasdk.edge.remote*), 100
- logout() (*cterasdk.client.host.CTERAHost* method), 52
- logout() (*cterasdk.core.login.Login* method), 70
- logout() (*cterasdk.edge.login.Login* method), 96
- Logs (class in *cterasdk.core.logs*), 70
- Logs (class in *cterasdk.edge.logs*), 96
- logs() (*cterasdk.edge.logs.Logs* method), 96
- LogTopic (class in *cterasdk.core.enum*), 67
- ls() (*cterasdk.core.files.browser.FileBrowser* method), 57
- ls() (*cterasdk.edge.files.browser.FileBrowser* static method), 79
- ls() (in module *cterasdk.core.files.ls*), 60
- lt() (*cterasdk.core.query.FilterBuilder* method), 72
- LU (*cterasdk.core.enum.CollaboratorType* attribute), 66
- LU (*cterasdk.edge.enum.PrincipalType* attribute), 90
- ## M
- Mail (class in *cterasdk.edge.mail*), 97
- make\_local\_files\_dir() (*cterasdk.object.Gateway.Gateway* static method), 115
- Manual (*cterasdk.edge.enum.ClientSideCaching* attribute), 88
- mkcol() (*cterasdk.client.cteraclient.CTERAClient* method), 51
- mkcol() (*cterasdk.client.host.CTERAHost* method), 52
- mkcol() (*cterasdk.client.http.HTTPClient* method), 53
- mkdir() (*cterasdk.core.cloudfs.CloudFS* method), 62
- mkdir() (*cterasdk.core.files.browser.FileBrowser* method), 57
- mkdir() (*cterasdk.edge.files.browser.FileBrowser* method), 79
- mkdir() (*cterasdk.lib.tempfile.TempfileServices* static method), 113
- mkdir() (in module *cterasdk.core.files.directory*), 60
- mkdir() (in module *cterasdk.edge.files.mkdir*), 79
- mkfg() (*cterasdk.core.cloudfs.CloudFS* method), 63

- mkfile() (*cterasdk.lib.tempfile.TempfileServices static method*), 113
- mklink() (*cterasdk.core.files.browser.FileBrowser method*), 58
- mklink() (*in module cterasdk.core.files.ln*), 60
- mkpath() (*cterasdk.core.files.browser.FileBrowser method*), 58
- mkpath() (*cterasdk.edge.files.browser.FileBrowser static method*), 79
- Mode (*class in cterasdk.edge.enum*), 90
- modify() (*cterasdk.edge.ftp.FTP method*), 95
- modify() (*cterasdk.edge.nfs.NFS method*), 98
- modify() (*cterasdk.edge.rsync.RSync method*), 100
- modify() (*cterasdk.edge.shares.Shares method*), 103
- modify() (*cterasdk.edge.smb.SMB method*), 105
- modify() (*cterasdk.edge.syslog.Syslog method*), 107
- modify() (*cterasdk.edge.users.Users method*), 110
- modify() (*cterasdk.edge.volumes.Volumes method*), 111
- Mounting (*cterasdk.edge.enum.VolumeStatus attribute*), 94
- move() (*cterasdk.core.files.browser.FileBrowser method*), 58
- move() (*in module cterasdk.core.files.mv*), 60
- move\_multi() (*cterasdk.core.files.browser.FileBrowser method*), 58
- move\_multi() (*in module cterasdk.core.files.mv*), 60
- ## N
- NA (*cterasdk.core.enum.FileAccessMode attribute*), 67
- NA (*cterasdk.edge.enum.FileAccessMode attribute*), 89
- name (*cterasdk.core.types.CloudFSFolderFindingHelper attribute*), 74
- name (*cterasdk.edge.types.ShareAccessControlEntry attribute*), 109
- name() (*cterasdk.core.files.path.CTERAPath method*), 61
- name() (*cterasdk.edge.files.path.CTERAPath method*), 80
- name\_attr (*cterasdk.core.devices.Devices attribute*), 65
- ne() (*cterasdk.core.query.FilterBuilder method*), 72
- Network (*class in cterasdk.edge.network*), 97
- NetworkHost (*class in cterasdk.client.host*), 53
- NFS (*class in cterasdk.edge.nfs*), 98
- no\_access() (*cterasdk.core.types.ShareRecipient method*), 76
- NoFolder (*cterasdk.edge.enum.BackupConfStatusID attribute*), 88
- NoFolder (*cterasdk.edge.enum.SyncStatus attribute*), 93
- NONE (*cterasdk.core.enum.PolicyType attribute*), 68
- none (*cterasdk.edge.support.DebugLevel attribute*), 106
- NOT\_EQUALS (*cterasdk.core.query.Restriction attribute*), 73
- NotFound, 83
- NotFound (*cterasdk.edge.backup.AttachRC attribute*), 82
- NOTICE (*cterasdk.core.enum.Severity attribute*), 69
- NOTICE (*cterasdk.edge.enum.Severity attribute*), 91
- NotInitialized (*cterasdk.edge.enum.BackupConfStatusID attribute*), 88
- NotInitialized (*cterasdk.edge.enum.SyncStatus attribute*), 93
- notLike() (*cterasdk.core.query.FilterBuilder method*), 72
- NTP (*class in cterasdk.edge.ntp*), 98
- ntp (*cterasdk.edge.support.DebugLevel attribute*), 106
- ## O
- OBJ (*cterasdk.convert.xml\_types.XMLTypes attribute*), 56
- Object (*class in cterasdk.common.object*), 55
- obtain\_ticket() (*in module cterasdk.edge.remote*), 100
- Off (*cterasdk.edge.enum.SyncStatus attribute*), 93
- OK (*cterasdk.edge.backup.AttachRC attribute*), 82
- OK (*cterasdk.edge.backup.CreateFolderRC attribute*), 82
- Ok (*cterasdk.edge.enum.VolumeStatus attribute*), 94
- on\_ssl\_error() (*cterasdk.client.http.HttpClientBase method*), 54
- on\_timeout() (*cterasdk.client.http.HttpClientBase static method*), 54
- OnlyAuthenticatedUsers (*cterasdk.edge.enum.Acl attribute*), 86
- Open (*cterasdk.edge.enum.TCPConnectRC attribute*), 93
- openfile() (*cterasdk.client.host.CTERAHost method*), 52
- OperationMode (*class in cterasdk.edge.enum*), 90
- orFilter() (*cterasdk.core.query.QueryParamBuilder method*), 72
- OriginType (*class in cterasdk.core.enum*), 67
- os() (*cterasdk.lib.platform.Platform method*), 112
- OutOfQuota (*cterasdk.edge.enum.SyncStatus attribute*), 93
- ownedBy() (*cterasdk.core.query.QueryParamBuilder method*), 72
- owner (*cterasdk.core.types.CloudFSFolderFindingHelper attribute*), 75
- ## P
- parent() (*cterasdk.core.files.path.CTERAPath method*), 61
- parent() (*cterasdk.edge.files.path.CTERAPath method*), 80
- ParseException, 55
- ParserException, 118

- ParseValue() (in module *cterasdk.convert.parse*), 56
- parts() (*cterasdk.core.files.path.CTERAPath* method), 61
- parts() (*cterasdk.edge.files.path.CTERAPath* method), 80
- perm (*cterasdk.edge.types.ShareAccessControlEntry* attribute), 109
- PermissionDenied (*cterasdk.edge.backup.AttachRC* attribute), 82
- PermissionDenied (*cterasdk.edge.backup.CreateFolderRC* attribute), 83
- pin() (*cterasdk.edge.cache.Cache* method), 83
- pin\_all() (*cterasdk.edge.cache.Cache* method), 83
- pin\_exclude() (*cterasdk.edge.cache.Cache* method), 83
- Plans (class in *cterasdk.core.plans*), 74
- Platform (class in *cterasdk.lib.platform*), 112
- PO (*cterasdk.core.enum.FileAccessMode* attribute), 67
- PolicyType (class in *cterasdk.core.enum*), 68
- port() (*cterasdk.client.host.NetworkHost* method), 53
- Portal (class in *cterasdk.object.Portal*), 116
- Portal (*cterasdk.core.enum.OriginType* attribute), 67
- PortalAccount (class in *cterasdk.core.types*), 75
- PortalAccountType (class in *cterasdk.core.enum*), 68
- Portals (class in *cterasdk.core.portals*), 70
- portals() (*cterasdk.core.reports.Reports* method), 73
- PortalType (class in *cterasdk.core.enum*), 68
- post() (*cterasdk.client.cteraclient.CTERAClient* method), 51
- post() (*cterasdk.client.host.CTERAHost* method), 52
- post() (*cterasdk.client.http.HTTPClient* method), 53
- Power (class in *cterasdk.edge.power*), 99
- prettify() (in module *cterasdk.transcript.transcribe*), 117
- preview\_only() (*cterasdk.core.types.ShareRecipient* method), 76
- principal\_type (*cterasdk.edge.types.ShareAccessControlEntry* attribute), 109
- principal\_type (*cterasdk.edge.types.UserGroupEntry* attribute), 109
- PrincipalType (class in *cterasdk.edge.enum*), 90
- process (*cterasdk.edge.support.DebugLevel* attribute), 106
- ProtectionLevel (class in *cterasdk.core.enum*), 68
- Public (*cterasdk.core.enum.ProtectionLevel* attribute), 68
- public\_info() (*cterasdk.object.Portal.Portal* method), 116
- put() (*cterasdk.client.cteraclient.CTERAClient* method), 51
- put() (*cterasdk.client.host.CTERAHost* method), 52
- put() (*cterasdk.client.http.HTTPClient* method), 53
- put() (*cterasdk.core.query.QueryParamBuilder* method), 72
- put() (*cterasdk.edge.query.QueryParamBuilder* method), 100
- put() (*cterasdk.exception.CTERAException* method), 118
- put() (*cterasdk.object.Portal.Portal* method), 116
- python\_version() (*cterasdk.lib.platform.Platform* method), 112
- PythonVersionException, 118
- ## Q
- query() (*cterasdk.object.Gateway.Gateway* method), 115
- query() (*cterasdk.object.Portal.Portal* method), 117
- query() (in module *cterasdk.core.query*), 73
- query() (in module *cterasdk.edge.query*), 100
- QueryParam (class in *cterasdk.edge.query*), 100
- QueryParamBuilder (class in *cterasdk.core.query*), 72
- QueryParamBuilder (class in *cterasdk.edge.query*), 100
- QueryParams (class in *cterasdk.core.query*), 72
- ## R
- RAID\_0 (*cterasdk.edge.enum.RAIDLevel* attribute), 91
- RAID\_1 (*cterasdk.edge.enum.RAIDLevel* attribute), 91
- RAID\_5 (*cterasdk.edge.enum.RAIDLevel* attribute), 91
- RAID\_6 (*cterasdk.edge.enum.RAIDLevel* attribute), 91
- RAIDLevel (class in *cterasdk.edge.enum*), 90
- read() (*cterasdk.client.http.HTTPResponse* method), 54
- read\_only() (*cterasdk.core.types.ShareRecipient* method), 76
- read\_write() (*cterasdk.core.types.ShareRecipient* method), 76
- ReadExtendedAttributes (*cterasdk.edge.enum.AuditEvents* attribute), 87
- ReadOnly (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- ReadOnlyAdmin (*cterasdk.core.enum.Role* attribute), 69
- ReadOnlyAdministrators (*cterasdk.edge.enum.LocalGroup* attribute), 89
- ReadWriteAdmin (*cterasdk.core.enum.Role* attribute), 69
- reboot() (*cterasdk.edge.power.Power* method), 99
- reconnect() (*cterasdk.edge.services.Services* method), 101
- Recoverable (*cterasdk.edge.backup.EncryptionMode* attribute), 83
- Recovering (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- refresh() (*cterasdk.edge.sync.Sync* method), 106

- register() (*cterasdk.lib.registry.Registry* method), 112
- register\_session() (*cterasdk.client.host.CTERAHost* method), 52
- Registry (class in *cterasdk.lib.registry*), 112
- RejectedByPolicy (*cterasdk.edge.enum.SyncStatus* attribute), 93
- Remote (*cterasdk.edge.session.SessionType* attribute), 102
- remote() (*cterasdk.edge.session.Session* method), 102
- remote() (in module *cterasdk.edge.uri*), 109
- remote\_access() (*cterasdk.edge.session.Session* method), 102
- remote\_access() (*cterasdk.object.Gateway.Gateway* method), 115
- remote\_access() (in module *cterasdk.edge.remote*), 100
- remote\_access() (in module *cterasdk.edge.uri*), 109
- remote\_command() (in module *cterasdk.core.remote*), 74
- remote\_from() (*cterasdk.edge.session.Session* method), 102
- RemoteDirectoryNotFound, 118
- RemoteFileSystemException, 118
- remove() (*cterasdk.lib.registry.Registry* method), 112
- remove\_acl() (*cterasdk.edge.shares.Shares* method), 104
- remove\_members() (*cterasdk.edge.groups.Groups* method), 96
- remove\_pin() (*cterasdk.edge.cache.Cache* method), 84
- remove\_share\_recipients() (*cterasdk.core.files.browser.FileBrowser* method), 58
- remove\_share\_recipients() (in module *cterasdk.core.files.collaboration*), 59
- RemoveShareAccessControlEntry (class in *cterasdk.edge.types*), 108
- rename() (*cterasdk.core.files.browser.FileBrowser* method), 58
- rename() (*cterasdk.lib.filesystem.FileSystem* method), 111
- rename() (in module *cterasdk.core.files.rename*), 61
- RenameException, 118
- Repairing (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- Reports (class in *cterasdk.core.reports*), 73
- Required (*cterasdk.edge.enum.CIFSPacketSigning* attribute), 88
- Reseller (*cterasdk.core.enum.PortalType* attribute), 68
- ReservedName, 60
- reset() (*cterasdk.edge.power.Power* method), 99
- Resizing (*cterasdk.edge.enum.VolumeStatus* attribute), 94
- resolve() (*cterasdk.edge.taskmgr.Task* static method), 107
- resolve() (*cterasdk.lib.tracker.StatusTracker* method), 113
- Restriction (class in *cterasdk.core.query*), 73
- rm() (*cterasdk.object.Gateway.Gateway* method), 115
- rmdir() (*cterasdk.lib.tempfile.TempfileServices* static method), 113
- rmfg() (*cterasdk.core.cloudfs.CloudFS* method), 63
- RO (*cterasdk.core.enum.FileAccessMode* attribute), 67
- RO (*cterasdk.edge.enum.FileAccessMode* attribute), 89
- Role (class in *cterasdk.core.enum*), 68
- RSync (class in *cterasdk.edge.rsycn*), 100
- rsync (*cterasdk.edge.support.DebugLevel* attribute), 106
- run\_command() (*cterasdk.edge.cli.CLI* method), 84
- run\_command() (*cterasdk.edge.shell.Shell* method), 104
- Running (*cterasdk.edge.enum.TaskStatus* attribute), 93
- Running (*cterasdk.edge.taskmgr.TaskStatusEnum* attribute), 108
- running() (*cterasdk.lib.tracker.StatusTracker* method), 113
- running() (in module *cterasdk.edge.taskmgr*), 108
- RW (*cterasdk.core.enum.FileAccessMode* attribute), 67
- RW (*cterasdk.edge.enum.FileAccessMode* attribute), 89
- ## S
- samba (*cterasdk.edge.support.DebugLevel* attribute), 106
- save() (*cterasdk.lib.filesystem.FileSystem* method), 111
- save\_cert\_from\_server() (*cterasdk.client.ssl.CertificateServices* static method), 54
- Scanning (*cterasdk.edge.enum.SyncStatus* attribute), 93
- scheme() (*cterasdk.client.host.NetworkHost* method), 53
- SearchType (class in *cterasdk.core.enum*), 69
- Secret (*cterasdk.edge.backup.EncryptionMode* attribute), 83
- SELECT (*cterasdk.core.enum.PolicyType* attribute), 68
- ServerAgent (*cterasdk.core.enum.DeviceType* attribute), 67
- Servers (class in *cterasdk.core.servers*), 74
- servers() (*cterasdk.core.devices.Devices* method), 65
- Services (class in *cterasdk.edge.services*), 101
- ServicesConnectionState (class in *cterasdk.edge.enum*), 91
- ServicesPortal (class in *cterasdk.object.Portal*), 117

- ServicesPortal (cterasdk.core.enum.Context attribute), 66
- ServiceUnavailable (cterasdk.edge.enum.SyncStatus attribute), 93
- Session (class in cterasdk.core.session), 74
- Session (class in cterasdk.edge.session), 102
- session() (cterasdk.client.host.CTERAHost method), 52
- session() (cterasdk.core.base\_command.BaseCommand method), 62
- session() (cterasdk.edge.base\_command.BaseCommand method), 83
- SessionBase (class in cterasdk.lib.session\_base), 113
- SessionConnection (class in cterasdk.edge.session), 102
- SessionStatus (class in cterasdk.lib.session\_base), 113
- SessionType (class in cterasdk.edge.session), 102
- SessionUser (class in cterasdk.lib.session\_base), 113
- set\_acl() (cterasdk.edge.shares.Shares method), 104
- set\_debug\_level() (cterasdk.edge.support.Support method), 106
- set\_hostname() (cterasdk.edge.config.Config method), 84
- set\_location() (cterasdk.edge.config.Config method), 85
- set\_packet\_signing() (cterasdk.edge.smb.SMB method), 105
- set\_session\_id() (cterasdk.client.cteraclient.CTERAClient method), 52
- set\_session\_id() (cterasdk.client.host.CTERAHost method), 52
- set\_session\_id() (cterasdk.client.http.HttpClientBase method), 54
- set\_share\_winacls() (cterasdk.edge.shares.Shares method), 104
- set\_static\_ipaddr() (cterasdk.edge.network.Network method), 97
- set\_static\_nameserver() (cterasdk.edge.network.Network method), 98
- set\_timezone() (cterasdk.edge.timezone.Timezone method), 108
- SetAppendValue() (in module cterasdk.convert.parse), 56
- setLevel() (cterasdk.config.Logging static method), 117
- setValue() (cterasdk.core.query.FilterBuilder method), 72
- Severity (class in cterasdk.core.enum), 69
- Severity (class in cterasdk.edge.enum), 91
- share() (cterasdk.core.files.browser.FileBrowser method), 58
- share() (in module cterasdk.core.files.collaboration), 59
- ShareAccessControlEntry (class in cterasdk.edge.types), 108
- ShareRecipient (class in cterasdk.core.types), 75
- Shares (class in cterasdk.edge.shares), 102
- Shell (class in cterasdk.edge.shell), 104
- should\_trust() (cterasdk.client.http.HttpClientBase method), 54
- ShouldSupportWinNtAcl (cterasdk.edge.enum.SyncStatus attribute), 93
- show() (cterasdk.client.host.CTERAHost method), 53
- show() (in module cterasdk.core.query), 73
- show() (in module cterasdk.edge.query), 100
- show\_multi() (cterasdk.client.host.CTERAHost method), 53
- show\_query() (cterasdk.object.Gateway.Gateway method), 115
- show\_query() (cterasdk.object.Portal.Portal method), 117
- shutdown() (cterasdk.edge.power.Power method), 99
- SMB (class in cterasdk.edge.smb), 104
- sortBy() (cterasdk.core.query.QueryParamBuilder method), 72
- split\_file\_directory() (cterasdk.lib.filesystem.FileSystem static method), 112
- StartParam (class in cterasdk.core.files.common), 59
- SSL (class in cterasdk.edge.ssl), 99
- SSLException, 118
- sso\_enabled() (cterasdk.edge.services.Services method), 101
- start() (cterasdk.edge.backup.Backup method), 82
- start\_local\_session() (cterasdk.lib.session\_base.SessionBase method), 113
- start\_remote\_session() (cterasdk.edge.session.Session method), 102
- startFrom() (cterasdk.core.query.QueryParamBuilder method), 72
- startFrom() (cterasdk.edge.query.QueryParamBuilder method), 100
- StatusTracker (class in cterasdk.lib.tracker), 113
- storage (cterasdk.edge.support.DebugLevel attribute), 106
- storage() (cterasdk.core.reports.Reports method), 73
- String (cterasdk.core.query.FilterType attribute), 72
- subscribe() (cterasdk.core.portals.Portals method), 71
- successful() (cterasdk.lib.tracker.StatusTracker method), 113

Support (class in *cterasdk.edge.support*), 106  
 Support (*cterasdk.core.enum.Role* attribute), 69  
 suspend() (*cterasdk.edge.backup.Backup* method), 82  
 suspend() (*cterasdk.edge.sync.Sync* method), 106  
 Sync (class in *cterasdk.edge.sync*), 106  
 Synced (*cterasdk.edge.enum.SyncStatus* attribute), 93  
 Syncing (*cterasdk.edge.enum.SyncStatus* attribute), 93  
 SyncStatus (class in *cterasdk.edge.enum*), 91  
 Syslog (class in *cterasdk.edge.syslog*), 107  
 System (*cterasdk.core.enum.LogTopic* attribute), 67

## T

TakingSnapshot (*cterasdk.edge.enum.SyncStatus* attribute), 93  
 Task (class in *cterasdk.edge.taskmgr*), 107  
 TaskError, 107  
 TaskStatus (class in *cterasdk.edge.enum*), 93  
 TaskStatusEnum (class in *cterasdk.edge.taskmgr*), 107  
 TCP (*cterasdk.edge.enum.IPProtocol* attribute), 89  
 tcp\_connect() (*cterasdk.edge.network.Network* method), 98  
 TCPConnectRC (class in *cterasdk.edge.enum*), 93  
 Team (*cterasdk.core.enum.PortalType* attribute), 68  
 Telnet (class in *cterasdk.edge.telnet*), 108  
 TempfileServices (class in *cterasdk.lib.tempfile*), 113  
 tenant() (*cterasdk.lib.session\_base.SessionBase* method), 113  
 tenants() (*cterasdk.core.portals.Portals* method), 71  
 terminate() (*cterasdk.lib.session\_base.SessionBase* method), 113  
 test() (*cterasdk.object.Gateway.Gateway* method), 115  
 test() (*cterasdk.object.Portal.Portal* method), 117  
 test() (in module *cterasdk.core.connection*), 63  
 test() (in module *cterasdk.edge.connection*), 85  
 test\_application() (in module *cterasdk.edge.connection*), 85  
 test\_conn() (*cterasdk.client.host.NetworkHost* method), 53  
 test\_network() (in module *cterasdk.core.connection*), 63  
 test\_network() (in module *cterasdk.edge.connection*), 85  
 textplain (*cterasdk.client.http.ContentType* attribute), 53  
 tid() (*cterasdk.edge.taskmgr.Task* static method), 107  
 Timezone (class in *cterasdk.edge.timezone*), 108  
 to\_server\_object() (*cterasdk.edge.types.ShareAccessControlEntry* method), 109  
 to\_server\_object() (*cterasdk.edge.types.UserGroupEntry* method),

109

tojsonstr() (in module *cterasdk.convert.format*), 55  
 toxml() (in module *cterasdk.convert.format*), 55  
 toxmlstr() (in module *cterasdk.convert.format*), 55  
 track() (*cterasdk.lib.tracker.StatusTracker* method), 113  
 track() (in module *cterasdk.lib.tracker*), 113  
 transcribe() (in module *cterasdk.transcript.transcribe*), 117  
 TraverseFolderExecuteFile (*cterasdk.edge.enum.AuditEvents* attribute), 87  
 trust() (*cterasdk.client.http.HttpClientBase* method), 54  
 type\_attr (*cterasdk.core.devices.Devices* attribute), 65

## U

UDP (*cterasdk.edge.enum.IPProtocol* attribute), 89  
 undelete() (*cterasdk.core.cloudfs.CloudFS* method), 63  
 undelete() (*cterasdk.core.files.brower.FileBrowser* method), 59  
 undelete() (*cterasdk.core.portals.Portals* method), 71  
 undelete() (in module *cterasdk.core.files.recover*), 61  
 undelete\_multi() (*cterasdk.core.files.brower.FileBrowser* method), 59  
 undelete\_multi() (in module *cterasdk.core.files.recover*), 61  
 union() (in module *cterasdk.core.union*), 76  
 Unknown (*cterasdk.edge.enum.VolumeStatus* attribute), 94  
 Unlicensed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 88  
 Unlicensed (*cterasdk.edge.enum.SyncStatus* attribute), 93  
 UNLIKE (*cterasdk.core.query.Restriction* attribute), 73  
 Unmounted (*cterasdk.edge.enum.VolumeStatus* attribute), 94  
 unpin\_all() (*cterasdk.edge.cache.Cache* method), 84  
 unshare() (*cterasdk.core.files.brower.FileBrowser* method), 59  
 unshare() (in module *cterasdk.core.files.collaboration*), 59  
 Unsubscribed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 88  
 unsuspend() (*cterasdk.edge.backup.Backup* method), 82  
 unsuspend() (*cterasdk.edge.sync.Sync* method), 106  
 update\_current\_tenant() (in module *cterasdk.core.decorator*), 63  
 update\_tenant() (*cterasdk.core.session.Session* method), 74

- upgrade() (*cterasdk.edge.firmware.Firmware method*), 95  
 UpgradingDataBase (*cterasdk.edge.enum.SyncStatus attribute*), 93  
 upload (*cterasdk.edge.support.DebugLevel attribute*), 106  
 upload() (*cterasdk.client.cteraclient.CTERAClient method*), 52  
 upload() (*cterasdk.client.host.CTERAHost method*), 53  
 upload() (*cterasdk.client.http.HTTPClient method*), 53  
 upload() (*cterasdk.core.files.browser.FileBrowser method*), 59  
 upload() (*cterasdk.edge.files.browser.FileBrowser method*), 79  
 upload() (*cterasdk.lib.file\_access\_base.FileAccessBase method*), 112  
 upload\_cert() (*cterasdk.edge.ssl.SSL method*), 99  
 UploadTaskStatus (*class in cterasdk.edge.firmware*), 95  
 urlencoded (*cterasdk.client.http.ContentType attribute*), 53  
 User (*cterasdk.core.enum.PortalAccountType attribute*), 68  
 UserAccount (*class in cterasdk.core.types*), 76  
 UserGroupEntry (*class in cterasdk.edge.types*), 109  
 Users (*class in cterasdk.core.users*), 76  
 Users (*class in cterasdk.edge.users*), 109  
 Users (*cterasdk.core.enum.SearchType attribute*), 69  
 UUID (*cterasdk.convert.xml\_types.XMLTypes attribute*), 56
- ## V
- VAL (*cterasdk.convert.xml\_types.XMLTypes attribute*), 56  
 validate\_directory() (*cterasdk.lib.filesystem.FileSystem method*), 112  
 Version (*class in cterasdk.lib.version*), 114  
 version() (*cterasdk.lib.filesystem.FileSystem static method*), 112  
 vGateway (*cterasdk.core.enum.DeviceType attribute*), 67  
 Volumes (*class in cterasdk.edge.volumes*), 110  
 VolumeStatus (*class in cterasdk.edge.enum*), 93  
 VolumeUnavailable (*cterasdk.edge.enum.SyncStatus attribute*), 93
- ## W
- wait() (*cterasdk.edge.power.Boot method*), 99  
 wait() (*cterasdk.edge.taskmgr.Task method*), 107  
 wait() (*in module cterasdk.edge.taskmgr*), 108  
 walk() (*cterasdk.core.files.browser.FileBrowser method*), 59  
 WARNING (*cterasdk.core.enum.Severity attribute*), 70  
 WARNING (*cterasdk.edge.enum.Severity attribute*), 91  
 warning (*cterasdk.edge.support.DebugLevel attribute*), 106  
 whoami() (*cterasdk.client.host.CTERAHost method*), 53  
 whoami() (*cterasdk.lib.session\_base.SessionBase method*), 113  
 WindowsNT (*cterasdk.edge.enum.Acl attribute*), 86  
 WorkstationAgent (*cterasdk.core.enum.DeviceType attribute*), 67  
 write() (*cterasdk.lib.filesystem.FileSystem static method*), 112  
 WriteAttributes (*cterasdk.edge.enum.AuditEvents attribute*), 87  
 WriteExtendedAttributes (*cterasdk.edge.enum.AuditEvents attribute*), 87  
 WrongPassword (*cterasdk.edge.enum.BackupConfStatusID attribute*), 88
- ## X
- XMLTypes (*class in cterasdk.convert.xml\_types*), 56
- ## Z
- Zones (*class in cterasdk.core.zones*), 77