
CTERA SDK for Python Documentation

CTERA Networks

Apr 06, 2020

Contents:

1	CTERA for Python	1
1.1	Documentation	1
1.2	Installation	1
1.3	Importing the Library	1
1.4	Building Documentation	2
1.5	Testing	2
2	User Guides	3
2.1	Global File System	3
2.1.1	Global Administration	3
2.1.1.1	Instantiate a Global Admin object	4
2.1.1.2	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	7
2.1.1.8	Devices	9
2.1.1.9	Zones	12
2.1.1.10	CloudFS	14
2.1.2	End User Portal	15
2.1.2.1	Instantiate a Services Portal object	16
2.1.3	File Browser	17
2.1.3.1	List	17
2.1.3.2	Download	17
2.1.3.3	Create Directory	17
2.1.3.4	Rename	18
2.1.3.5	Delete	18
2.1.3.6	Recover	18
2.1.3.7	Copy	19
2.1.3.8	Move	19
2.1.3.9	Create Public Link	19
2.2	Edge Filer	20
2.2.1	Gateway	20
2.2.1.1	Instantiate a Gateway object	21
2.2.2	File Browser	41

2.2.2.1	Obtaining Access to the Gateway's File System	41
2.3	Agent	42
2.4	Miscellaneous	43
2.4.1	Logging	43
2.4.1.1	Redirecting the log to a file	43
2.4.1.2	Disabling the Logger	43
2.4.1.3	Changing the Log Level	43
2.4.2	Formatting	43
3	cterasdk package	45
3.1	Subpackages	45
3.1.1	cterasdk.client package	45
3.1.1.1	Submodules	45
3.1.2	cterasdk.common package	48
3.1.2.1	Submodules	48
3.1.3	cterasdk.convert package	48
3.1.3.1	Submodules	48
3.1.4	cterasdk.core package	50
3.1.4.1	Subpackages	50
3.1.4.2	Submodules	53
3.1.5	cterasdk.edge package	68
3.1.5.1	Subpackages	68
3.1.5.2	Submodules	69
3.1.6	cterasdk.lib package	100
3.1.6.1	Submodules	100
3.1.7	cterasdk.object package	102
3.1.7.1	Submodules	102
3.1.8	cterasdk.transcript package	105
3.1.8.1	Submodules	105
3.2	Submodules	105
3.2.1	cterasdk.config module	105
3.2.2	cterasdk.exception module	105
4	Indices and tables	107
5	Help Us Improve the Docs <3	109
	Python Module Index	111
	Index	113

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*
 - * *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*
 - * *Create a Folder Group*
 - * *Delete a Folder Group*
 - * *Create a Cloud Drive Folder*
 - * *Delete a Cloud Drive Folder*
 - * *Recover a Cloud Drive Folder*

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.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)`

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

Return str A relative url path to the Team Portal

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

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

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

Local Users

Users.list_local_users (*include=None*)

List all local users

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

Returns Iterator for all local users

Return type *cterasdk.lib.iterator.Iterator*

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

for user in users:

    print(user.name)

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

for user in users:

    print(user)
```

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

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

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

`Users.delete(name)`

Delete an existing user

Parameters **name** (*str*) – The user name to delete

```
"""Delete a local user"""
admin.users.delete('bruce')
```

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, tenant*)

Generate device activation code

Parameters

- **username** (*str*) – User name used for activation
- **tenant** (*str*) – Tenant name used for activation

Returns Portal Activation Code

Return type *str*

```
"""Generate a device activation code"""

code = admin.activation.generate_code('bruce') # will look for 'bruce' in the current_
↳tenant

code = admin.activation.generate_code('batman', 'gotham') # will look 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
"""
```

(continues on next page)

(continued from previous page)

```

"""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

Create a Folder Group

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
```

Delete a Folder Group

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

Remove a Folder Group

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

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

Create a Cloud Drive Folder

CloudFS.**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

```

"""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)

```

Delete a Cloud Drive Folder

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)

```

Recover a Cloud Drive Folder

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*

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)`

Download a file

Parameters `path` (*str*) – Path of the file to download

```
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 length 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"""
```

(continues on next page)

(continued from previous page)

```
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.

2.2 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*
- * *Power Management*
- * *Support*
 - *Support Report*
 - *Debug*
 - *Telnet Access*

2.2.1.1 Instantiate a Gateway object

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

Main class operating on a Gateway

Variables

- **config** (cterasdk.edge.config.Config) – Object holding the Gateway Configuration APIs
- **network** (cterasdk.edge.network.Network) – Object holding the Gateway Network APIs
- **licenses** (cterasdk.edge.licenses.Licenses) – Object holding the Gateway Licenses APIs
- **services** (cterasdk.edge.services.Services) – Object holding the Gateway Services APIs
- **directoryservice** (cterasdk.edge.directoryservice.DirectoryService) – Object holding the Gateway Active Directory APIs
- **telnet** (cterasdk.edge.telnet.Telnet) – Object holding the Gateway Telnet APIs
- **syslog** (cterasdk.edge.syslog.Syslog) – Object holding the Gateway Syslog APIs
- **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
- **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

`__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

```
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', 'G3n3ralZ0d!')
```

`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)
```

(continues on next page)

(continued from previous page)

```

"""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'])

print(device.config.cloudsync.cloudExtender.operationMode)
print(device.proc.cloudsync.serviceStatus.uploadingFiles)

```

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

```

"""Disable the first time wizard"""

filer.put('/config/gui/openFirstTimeWizard', False)

"""Turn off FTP access on all shares"""

shares = filer.get('/config/fileservices/share')

for share in shares:

    share.exportToFTP = False

    filer.put('/config/fileservices/share/' + share.name, share)

```

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

```

"""Execute the file-eviction process"""

filer.execute('/config/cloudsync', 'forceExecuteEvictor') # doesn't require a param

"""Reboot the Gateway"""

filer.execute('/statuc/device', 'reboot') # doesn't require a param

"""TCP Connect"""

param = Object()

param.address = 'chopin.ctera.com'

```

(continues on next page)

(continued from previous page)

```
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'

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()
```

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)
```

(continues on next page)

(continued from previous page)

```

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
"""

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

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

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

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

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

Parameters

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

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

Groups

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

Parameters

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

```
"""Add Bruce Wayne to the local Administrators group"""
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')])  
  
"""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!l3')  
  
"""Connect to the EMEA Organizational Unit"""  
filer.directoryservice.connect('ctera.local', 'administrator', 'B4tMob!l3', '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.rsycn.disable()
```

Windows File Sharing (CIFS/SMB)

`SMB.enable()`
Enable SMB

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

`SMB.disable()`
Disble 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()`

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*)
Download a file

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

```
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)
```

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 cterasdk log to a file by setting the environment variable **CTERASDK_LOG_FILE**

2.4.1.2 Disabling the Logger

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

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

2.4.1.3 Changing the Log Level

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

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

Log Levels

The available log levels are:

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

2.4.2 Formatting

The following formatting functions are included in this library:

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

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`

Returns JSON string of the object

Return type str

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

`cterasdk.convert.format.toxmlstr(obj, pretty_print=False)`

Convert a Python object to an XML string

Parameters

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

Returns XML string of the object

Return type str

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

3.1 Subpackages

3.1.1 cterasdk.client package

3.1.1.1 Submodules

cterasdk.client.cteraclient module

```
class cterasdk.client.cteraclient.CTERAClient
    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)
    mkcol (baseurl, path)
    post (baseurl, path, data)
    put (baseurl, path, data)
```

```
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.

```
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 ()
```

```
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 cteraskd.client.host.**NetworkHost** (*host, port, https*)

Bases: object

baseurl()

host()

https()

port()

scheme()

test_conn()

cteraskd.client.host.**authenticated** (*function*)

cteraskd.client.http module

class cteraskd.client.http.**ContentType**

Bases: object

textplain = {'Content-Type': 'text/plain'}

urlencoded = {'Content-Type': 'application/x-www-form-urlencoded'}

class cteraskd.client.http.**HTTPClient**

Bases: *cteraskd.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 cteraskd.client.http.**HTTPException** (*http_error*)

Bases: Exception

class cteraskd.client.http.**HTTPResponse** (*response*)

Bases: object

getcode()

geturl()

read()

class cteraskd.client.http.**HttpClientBase**

Bases: object

dispatch (*ctera_request*)

on_ssl_error (*request*)

static on_timeout (*attempt*)

```
    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.item module

```
class cteraskd.common.item.Item
    Bases: object
```

cteraskd.common.object module

```
class cteraskd.common.object.Object
    Bases: object
```

3.1.3 cteraskd.convert package

3.1.3.1 Submodules

cterasdk.convert.exception module

exception cteraskd.convert.exception.ParseException

Bases: Exception

cterasdk.convert.format module

cterasdk.convert.format.CreateElement (*parent, tag*)

cterasdk.convert.format.tojsonstr (*obj, pretty_print=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

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 cteraskd.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

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*)

Download a file

Parameters **path** (*str*) – Path of the file to download

download_as_zip (*cloud_directory, files*)

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

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*)

rename (*path, name*)

Rename a file

Parameters

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

undelete (*path*)

Restore a previously deleted file or directory

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

undelete_multi (**args*)

Restore previously deleted multiple files and/or directories

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

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.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.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.mlink (ctera_host, path, access, expire_in)
```

cterasdk.core.files.ls module

`cterasdk.core.files.ls.list_dir(ctera_host, param)`

`cterasdk.core.files.ls.ls(ctera_host, path)`

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, tenant*)

Generate device activation code

Parameters

- **username** (*str*) – User name used for activation
- **tenant** (*str*) – Tenant name used for activation

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*)

List folder groups :param str,optional include: List of fields to retrieve, defaults to ['name', 'owner']

:returns: Iterator for all folder groups

list_folders (*include=None, deleted=False*)

List cloud drive folders :param str,optional include: List of fields to retrieve, defaults to ['name', 'group', 'owner'] :param str,optional deleted: Retrieve deleted folders :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*)

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

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.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 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.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.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*)
 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

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

delete (*name*)

Delete an existing tenant

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

tenants (*include_deleted=False*)

Get all tenants

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

undelele (*name*)

Undelete a previously deleted tenant

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

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.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

    static fromValue (value)

class cterasdk.core.query.QueryParamBuilder
    Bases: object

    addFilter (query_filter)

    allPortals (allPortals)

    build ()

    countLimit (countLimit)

    include (include)

    include_classname ()

    orFilter (orFilter)

    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.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.common.object.Object*

```
activate(tenant, user, role)
```

```
authenticated()
```

```
global_admin()
```

```
initialize()
```

```
initializing()
```

```
tenant()
```

```
update_tenant(current_tenant)
```

```
user_name()
```

```

    whoami ()

class cteraskd.core.session.SessionStatus
    Bases: object

    Active = 'Active'

    Inactive = 'Inactive'

    Initializing = 'Initializing'

cteraskd.core.session.activate (Portal)
cteraskd.core.session.inactive_session (Portal)
cteraskd.core.session.obtain_tenant (CTERAHost)
cteraskd.core.session.obtain_user (CTERAHost)
cteraskd.core.session.terminate (Portal)

```

cteraskd.core.types module

```

class cteraskd.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 cteraskd.core.types.GroupAccount (name, directory=None)
    Bases: cteraskd.core.types.PortalAccount

    account_type
        The Portal Account Type

        Return cteraskd.core.enum.PortalAccountType The Portal Account Type

class cteraskd.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 cteraskd.core.enum.PortalAccountType The Portal Account Type

    is_local
        Is the account local

        Return bool True if the account if local, otherwise False

class cteraskd.core.types.UserAccount (name, directory=None)
    Bases: cteraskd.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*)

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

delete (*name*)

Delete an existing user

Parameters **name** (*str*) – The user name to delete

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*)

Download a file

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

download_as_zip (*cloud_directory*, *files*)

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

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.**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.**ClockOutOfSync**

Bases: *cterasdk.exception.CTERAException*

Clock Out of Sync exception

class cterask.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 cterask.edge.backup.**IncorrectPassphrase**

Bases: *cterask.exception.CTERAException*

Incorrect Passphrase exception

exception cterask.edge.backup.**NotFound** (*message=None, instance=None, **kwargs*)

Bases: *cterask.exception.CTERAException*

Not found exception

cterask.edge.base_command module

class cterask.edge.base_command.**BaseCommand** (*gateway*)

Bases: object

Base class for all Gateway API classes

session ()

cterask.edge.cache module

class cterask.edge.cache.**Cache** (*gateway*)

Bases: *cterask.edge.base_command.BaseCommand*

Gateway cache configuration

disable ()

Disable caching

enable ()

Enable caching

force_eviction ()

Force eviction

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

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 cteraskd.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 cteraskd.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 cteraskd.edge.enum.FileAccessMode
    Bases: object
    File Access Mode
        Variables
            • RW (str) – Read Write
            • RO (str) – ReadOnly
            • NA (str) – None
    NA = 'None'
    RO = 'ReadOnly'
    RW = 'ReadWrite'
class cteraskd.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, anony-
        mous_ftp_folder=None, banner_message=None, max_connections_per_ip=None, re-
        quire_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.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,optinal 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.power module

class `cterasdk.edge.power.Boot` (*gateway, retries=60, seconds=5*)

Bases: `object`

wait()

class `cterasdk.edge.power.Power` (*gateway*)

Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Power APIs

reboot (*wait=False*)

Reboot the Gateway

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

reset (*wait=False*)

Reset the Gateway setting

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

shutdown()

Shutdown the Gateway

cterasdk.edge.query module

```
class cterasdk.edge.query.QueryParam
    Bases: cterasdk.common.object.Object

    include_classname ()

    increment ()

class cterasdk.edge.query.QueryParamBuilder
    Bases: object

    allPortals (allPortals)

    build ()

    countLimit (countLimit)

    include (include)

    include_classname ()

    put (key, value)

    sortBy (sortBy)

    startFrom (startFrom)

cterasdk.edge.query.query (CTERAHost, path, key, value)
cterasdk.edge.query.show (CTERAHost, path, key, value)
```

cterasdk.edge.remote module

```
cterasdk.edge.remote.login (Gateway, ticket)
cterasdk.edge.remote.obtain_ticket (Portal, device_name)
cterasdk.edge.remote.remote_access (Gateway, Portal)
```

cterasdk.edge.rsync module

```
class cterasdk.edge.rsync.RSync (gateway)
    Bases: cterasdk.edge.base_command.BaseCommand

    Gateway RSync configuration

    disable ()
        Disable FTP

    enable ()
        Enable FTP

    get_configuration ()
        Get the current RSync configuration

        Return cterasdk.common.object.Object

    is_disabled ()
        Check if the Rsync server is disabled

    modify (port=None, max_connections=None)
        Modify the RSync Configuration. Parameters that are not passed will not be affected
```

Parameters

- **port** (*int, optional*) – RSync Port
- **max_connections** (*int, optional*) – Maximum Connections

cterasdk.edge.services module

class `cterasdk.edge.services.Services` (*gateway*)
 Bases: `cterasdk.edge.base_command.BaseCommand`

Gateway Cloud Services configuration APIs

activate (*server, user, code, ctera_license='EV16'*)
 Activate the gateway using an activation code

Parameters

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

connect (*server, user, password, ctera_license='EV16'*)
 Connect to a Portal.

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

Parameters

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

connected()
 Check if the Edge Filer is connected to CTERA Portal

disable_sso()
 Disable SSO connection

disconnect()
 Disconnect from the Portal

enable_sso()
 Enable SSO connection

get_status()
 Retrieve the cloud services connection status

reconnect()
 Reconnect to the Portal

sso_enabled()

Is SSO connection enabled

Return bool True if SSO connection is enabled, else False

cterasdk.edge.session module

class cterasdk.edge.session.**ActiveSession**(*host, session_type, user*)

Bases: *cterasdk.edge.session.Session*

username()

class cterasdk.edge.session.**InactiveSession**(*host*)

Bases: *cterasdk.edge.session.Session*

class cterasdk.edge.session.**LocalSession**(*host, user*)

Bases: *cterasdk.edge.session.ActiveSession*

class cterasdk.edge.session.**RemoteSession**(*host, user, remote_from, tenant*)

Bases: *cterasdk.edge.session.ActiveSession*

disable_remote_access()

enable_remote_access()

remote_access()

remote_from()

tenant()

class cterasdk.edge.session.**Session**(*host, session_type, status*)

Bases: *cterasdk.common.object.Object*

authenticated()

local()

remote()

target_host()

class cterasdk.edge.session.**SessionStatus**

Bases: *object*

Active = 'Active'

Inactive = 'Inactive'

class cterasdk.edge.session.**SessionType**

Bases: *object*

Local = 'local'

Remote = 'remote'

cterasdk.edge.session.**inactive_session**(*Gateway*)

cterasdk.edge.session.**start_local_session**(*Gateway, host, user*)

cterasdk.edge.session.**start_remote_session**(*Gateway, Portal*)

cterasdk.edge.session.**terminate**(*Gateway*)

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 cterask.edge.support.Support(gateway)
    Bases: cterask.edge.base_command.BaseCommand
    Gateway Support APIs
    get_support_report()
        Download support report
    set_debug_level(*levels)
        Set the debug level

```

cterask.edge.sync module

```

class cterask.edge.sync.Sync(gateway)
    Bases: cterask.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

```

cterask.edge.syslog module

```

class cterask.edge.syslog.Syslog(gateway)
    Bases: cterask.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** (*cterask.edge.enum.IPProtocol, optional*) – Syslog server communication protocol, defaults to cterask.edge.enum.IPProtocol.UDP

- **min_severity** (*cterasdk.edge.enum.Severity, optional*) – Minimal log severity to fetch, defaults to *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 cteraskd.lib package

3.1.6.1 Submodules

cteraskd.lib.cmd module

```
class cteraskd.lib.cmd.Command(cmd, *args)  
    Bases: object
```

cteraskd.lib.consent module

```
cteraskd.lib.consent.ask(question)
```

cteraskd.lib.filesystem module

```
class cteraskd.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)  
  
    validate_directory(dirpath)  
  
    static version(filename, version)  
  
    static write(filepath, handle)
```

cteraskd.lib.file_access_base module

```
class cteraskd.lib.file_access_base.FileAccessBase(ctera_host)  
    Bases: abc.ABC  
  
    download(path)  
  
    download_as_zip(cloud_directory, files)  
  
    upload(local_file, dest_path)
```

cteraskd.lib.iterator module

```
class cteraskd.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.tempfile module

```
class cterasdk.lib.tempfile.TempfileServices
    Bases: object

    static mkdir()

    static mkfile(prefix, suffix)

    static rmdir()
```

cterasdk.lib.tracker module

```
exception cterasdk.lib.tracker.ErrorStatus(status)
    Bases: cterasdk.exception.CTERAException

class cterasdk.lib.tracker.StatusTracker(CTERAHost, ref, success, progress, transient,
                                          failure, retries, seconds)
    Bases: object

    failed()

    increment()

    resolve()

    running()

    successful()

    track()

cterasdk.lib.tracker.track(CTERAHost, ref, success, progress, transient, failure, retries=300, seconds=1)
```

cterasdk.lib.version module

```
class cterasdk.lib.version.Version
    Bases: object
    as_header()
    static instance()
```

3.1.7 cterasdk.object package

3.1.7.1 Submodules

cterasdk.object.Agent module

```
class cterasdk.object.Agent.Agent (host, port=80, https=False, Portal=None)
    Bases: cterasdk.client.host.CTERAHost
    Main class operating on a Agent
    base_api_url
```

cterasdk.object.Gateway module

```
class cterasdk.object.Gateway.Gateway (host, port=80, 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
- **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

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 cterasdk.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 cterasdk.object.Portal.**Portal** (*host, port, https*)

Bases: *cterasdk.client.host.CTERAHost*

Parent class for communicating with the Portal through either GlobalAdmin or ServicesPortal

Variables

- **users** (*cterasdk.core.users.Users*) – Object holding the Portal user APIs
- **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*)

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 cterask.object.Portal.**ServicesPortal** (*host, port=443, https=True*)

Bases: *cterask.object.Portal.Portal*

Main class for Service operations on a Portal

context

file_browser_base_path

3.1.8 cterask.transcript package

3.1.8.1 Submodules

cterask.transcript.transcribe module

cterask.transcript.transcribe.**prettyfy** (*data*)

cterask.transcript.transcribe.**transcribe** (*request, response=None*)

3.2 Submodules

3.2.1 cterask.config module

class cterask.config.**Logging**

Bases: object

static **df** ()

static **disable** ()

static **enable** ()

static **fmt** ()

static **get** ()

static **setLevel** (*level*)

3.2.2 cterask.exception module

exception cterask.exception.**CTERAClientException** (*message=None, instance=None, **kwargs*)

Bases: *cterask.exception.CTERAException*

exception cterask.exception.**CTERAConnectionError** (*message, instance, host, port, protocol, **kwargs*)

Bases: *cterask.exception.CTERAException*

exception cterask.exception.**CTERAException** (*message=None, instance=None, **kwargs*)

Bases: Exception

join (*instance=None*)

```
put (**kwargs)

exception cterask.exception.ConnectionTimeout (message, seconds, **kwargs)
    Bases: cterask.exception.CTERAException

exception cterask.exception.ConsentException
    Bases: cterask.exception.CTERAException

exception cterask.exception.ExhaustedException (retries, timeout)
    Bases: cterask.exception.ConnectionTimeout

exception cterask.exception.FileSystemException (message=None, instance=None,
                                                **kwargs)
    Bases: cterask.exception.CTERAException

exception cterask.exception.HostUnreachable (instance, host, port, protocol)
    Bases: cterask.exception.CTERAConnectionError

exception cterask.exception.InputError (message, expression, options)
    Bases: cterask.exception.CTERAException

exception cterask.exception.LocalDirectoryNotFound (path)
    Bases: cterask.exception.FileSystemException

exception cterask.exception.LocalFileNotFound (path)
    Bases: cterask.exception.FileSystemException

exception cterask.exception.ParserException (fmt, payload)
    Bases: cterask.exception.CTERAException

exception cterask.exception.PythonVersionException (version)
    Bases: cterask.exception.CTERAException

exception cterask.exception.RemoteDirectoryNotFound (path)
    Bases: cterask.exception.RemoteFileSystemException

exception cterask.exception.RemoteFileSystemException (message=None, instance=None, **kwargs)
    Bases: cterask.exception.CTERAException

exception cterask.exception.RenameException (dirpath, src, dst)
    Bases: cterask.exception.FileSystemException

exception cterask.exception.SSLErrorException (host, port, reason)
    Bases: cterask.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`, 45
- `cterasdk.client`, 45
 - `cterasdk.client.cteraclient`, 45
 - `cterasdk.client.host`, 46
 - `cterasdk.client.http`, 47
 - `cterasdk.client.ssl`, 48
- `cterasdk.common`, 48
 - `cterasdk.common.item`, 48
 - `cterasdk.common.object`, 48
- `cterasdk.config`, 105
- `cterasdk.convert`, 48
 - `cterasdk.convert.exception`, 49
 - `cterasdk.convert.format`, 49
 - `cterasdk.convert.parse`, 49
 - `cterasdk.convert.xml_types`, 49
- `cterasdk.core`, 50
 - `cterasdk.core.activation`, 53
 - `cterasdk.core.base_command`, 54
 - `cterasdk.core.cloudfs`, 54
 - `cterasdk.core.connection`, 55
 - `cterasdk.core.decorator`, 55
 - `cterasdk.core.devices`, 55
 - `cterasdk.core.directoryservice`, 57
 - `cterasdk.core.enum`, 57
 - `cterasdk.core.files`, 50
 - `cterasdk.core.files.browser`, 50
 - `cterasdk.core.files.common`, 52
 - `cterasdk.core.files.cp`, 52
 - `cterasdk.core.files.directory`, 52
 - `cterasdk.core.files.file_access`, 52
 - `cterasdk.core.files.ln`, 52
 - `cterasdk.core.files.ls`, 53
 - `cterasdk.core.files.mv`, 53
 - `cterasdk.core.files.path`, 53
 - `cterasdk.core.files.recover`, 53
 - `cterasdk.core.files.rename`, 53
 - `cterasdk.core.files.rm`, 53
 - `cterasdk.core.login`, 61
 - `cterasdk.core.logs`, 61
 - `cterasdk.core.portals`, 61
 - `cterasdk.core.query`, 63
 - `cterasdk.core.remote`, 64
 - `cterasdk.core.reports`, 62
 - `cterasdk.core.servers`, 64
 - `cterasdk.core.session`, 64
 - `cterasdk.core.types`, 65
 - `cterasdk.core.union`, 66
 - `cterasdk.core.users`, 66
 - `cterasdk.core.zones`, 67
- `cterasdk.edge`, 68
 - `cterasdk.edge.afp`, 69
 - `cterasdk.edge.aio`, 70
 - `cterasdk.edge.array`, 70
 - `cterasdk.edge.audit`, 70
 - `cterasdk.edge.backup`, 71
 - `cterasdk.edge.base_command`, 72
 - `cterasdk.edge.cache`, 72
 - `cterasdk.edge.cli`, 73
 - `cterasdk.edge.config`, 73
 - `cterasdk.edge.connection`, 74
 - `cterasdk.edge.decorator`, 74
 - `cterasdk.edge.directoryservice`, 74
 - `cterasdk.edge.drive`, 75
 - `cterasdk.edge.enum`, 75
 - `cterasdk.edge.files`, 68
 - `cterasdk.edge.files.browser`, 68
 - `cterasdk.edge.files.file_access`, 69
 - `cterasdk.edge.files.mkdir`, 69
 - `cterasdk.edge.files.path`, 69
 - `cterasdk.edge.files.rm`, 69
 - `cterasdk.edge.ftp`, 83
 - `cterasdk.edge.groups`, 84
 - `cterasdk.edge.licenses`, 84
 - `cterasdk.edge.login`, 85
 - `cterasdk.edge.logs`, 85
 - `cterasdk.edge.mail`, 85
 - `cterasdk.edge.network`, 86
 - `cterasdk.edge.nfs`, 86

- `cterasdk.edge.ntp`, 87
- `cterasdk.edge.power`, 87
- `cterasdk.edge.query`, 88
- `cterasdk.edge.remote`, 88
- `cterasdk.edge.rsync`, 88
- `cterasdk.edge.services`, 89
- `cterasdk.edge.session`, 90
- `cterasdk.edge.shares`, 91
- `cterasdk.edge.shell`, 93
- `cterasdk.edge.smb`, 93
- `cterasdk.edge.support`, 94
- `cterasdk.edge.sync`, 95
- `cterasdk.edge.syslog`, 95
- `cterasdk.edge.taskmgr`, 96
- `cterasdk.edge.telnet`, 96
- `cterasdk.edge.timezone`, 97
- `cterasdk.edge.types`, 97
- `cterasdk.edge.uri`, 98
- `cterasdk.edge.users`, 98
- `cterasdk.edge.volumes`, 99
- `cterasdk.exception`, 105
- `cterasdk.lib`, 100
 - `cterasdk.lib.cmd`, 100
 - `cterasdk.lib.consent`, 100
 - `cterasdk.lib.file_access_base`, 100
 - `cterasdk.lib.filesystem`, 100
 - `cterasdk.lib.iterator`, 100
 - `cterasdk.lib.platform`, 101
 - `cterasdk.lib.registry`, 101
 - `cterasdk.lib.tempfile`, 101
 - `cterasdk.lib.tracker`, 101
 - `cterasdk.lib.version`, 102
- `cterasdk.object`, 102
 - `cterasdk.object.Agent`, 102
 - `cterasdk.object.Gateway`, 102
 - `cterasdk.object.Portal`, 104
- `cterasdk.transcript`, 105
 - `cterasdk.transcript.transcribe`, 105

A

- aapi (*cterasdk.edge.support.DebugLevel* attribute), 94
- Access (*cterasdk.core.enum.LogTopic* attribute), 58
- account_type (*cterasdk.core.types.GroupAccount* attribute), 65
- account_type (*cterasdk.core.types.PortalAccount* attribute), 65
- account_type (*cterasdk.core.types.UserAccount* attribute), 65
- Acl (class in *cterasdk.edge.enum*), 75
- ActionResourcesParam (class in *cterasdk.core.files.common*), 52
- activate() (*cterasdk.core.session.Session* method), 64
- activate() (*cterasdk.edge.services.Services* method), 89
- activate() (in module *cterasdk.core.session*), 65
- Activation (class in *cterasdk.core.activation*), 53
- Active (*cterasdk.core.session.SessionStatus* attribute), 65
- Active (*cterasdk.edge.session.SessionStatus* attribute), 90
- ActiveSession (class in *cterasdk.edge.session*), 90
- add() (*cterasdk.client.host.CTERAHost* method), 46
- add() (*cterasdk.core.files.common.ActionResourcesParam* method), 52
- add() (*cterasdk.core.portals.Portals* method), 61
- add() (*cterasdk.core.users.Users* method), 66
- add() (*cterasdk.core.zones.Zones* method), 67
- add() (*cterasdk.edge.array.Array* method), 70
- add() (*cterasdk.edge.shares.Shares* method), 91
- add() (*cterasdk.edge.users.Users* method), 98
- add() (*cterasdk.edge.volumes.Volumes* method), 99
- add_acl() (*cterasdk.edge.shares.Shares* method), 91
- add_devices() (*cterasdk.core.zones.Zones* method), 67
- add_first_user() (*cterasdk.edge.users.Users* method), 98
- add_folders() (*cterasdk.core.zones.Zones* method), 67
- add_members() (*cterasdk.edge.groups.Groups* method), 84
- add_trusted_cert() (*cterasdk.client.ssl.CertificateServices* static method), 48
- addFilter() (*cterasdk.core.query.QueryParamBuilder* method), 63
- admin (*cterasdk.core.enum.Context* attribute), 58
- Administrators (*cterasdk.edge.enum.LocalGroup* attribute), 78
- advanced_mapping() (*cterasdk.edge.directoryservice.DirectoryService* method), 74
- AFP (class in *cterasdk.edge.afp*), 69
- after() (*cterasdk.core.query.FilterBuilder* method), 63
- Agent (class in *cterasdk.object.Agent*), 102
- Agents (*cterasdk.core.enum.DeviceType* attribute), 58
- agents() (*cterasdk.core.devices.Devices* method), 55
- AIO (class in *cterasdk.edge.aio*), 70
- ALERT (*cterasdk.core.enum.Severity* attribute), 60
- ALERT (*cterasdk.edge.enum.Severity* attribute), 80
- alert (*cterasdk.edge.support.DebugLevel* attribute), 94
- ALL (*cterasdk.core.enum.PolicyType* attribute), 59
- allPortals() (*cterasdk.core.query.QueryParamBuilder* method), 63
- allPortals() (*cterasdk.edge.query.QueryParamBuilder* method), 88
- api() (in module *cterasdk.edge.uri*), 98
- apply() (*cterasdk.edge.licenses.Licenses* method), 84
- apps (*cterasdk.edge.support.DebugLevel* attribute), 94
- arch() (*cterasdk.lib.platform.Platform* method), 101
- Array (class in *cterasdk.edge.array*), 70
- as_header() (*cterasdk.lib.version.Version* method), 102
- ask() (in module *cterasdk.lib.consent*), 100
- ATT (*cterasdk.convert.xml_types.XMLTypes* attribute), 49
- Attached (*cterasdk.edge.enum.BackupConfStatusID* attribute), 76

AttachEncrypted, 71
 Attaching (*cterasdk.edge.enum.BackupConfStatusID* attribute), 76
 Audit (*class in cterasdk.edge.audit*), 70
 Audit (*cterasdk.core.enum.LogTopic* attribute), 58
 AuditEvents (*class in cterasdk.edge.enum*), 75
 auth (*cterasdk.edge.support.DebugLevel* attribute), 94
 authenticated() (*cterasdk.core.session.Session* method), 64
 authenticated() (*cterasdk.edge.session.Session* method), 90
 authenticated() (*in module cterasdk.client.host*), 47
 authenticated() (*in module cterasdk.edge.decorator*), 74
 av (*cterasdk.edge.support.DebugLevel* attribute), 94

B

Backup (*class in cterasdk.edge.backup*), 71
 backup (*cterasdk.edge.support.DebugLevel* attribute), 94
 BackupConfStatusID (*class in cterasdk.edge.enum*), 76
 base_api_url (*cterasdk.client.host.CTERAHost* attribute), 46
 base_api_url (*cterasdk.object.Agent.Agent* attribute), 102
 base_api_url (*cterasdk.object.Gateway.Gateway* attribute), 103
 base_api_url (*cterasdk.object.Portal.Portal* attribute), 104
 base_file_url (*cterasdk.client.host.CTERAHost* attribute), 46
 base_file_url (*cterasdk.object.Gateway.Gateway* attribute), 103
 base_file_url (*cterasdk.object.Portal.Portal* attribute), 104
 base_portal_url (*cterasdk.object.Portal.Portal* attribute), 104
 BaseCommand (*class in cterasdk.core.base_command*), 54
 BaseCommand (*class in cterasdk.edge.base_command*), 72
 baseurl() (*cterasdk.client.host.NetworkHost* method), 47
 before() (*cterasdk.core.query.FilterBuilder* method), 63
 block_files() (*cterasdk.edge.shares.Shares* method), 91
 Boot (*class in cterasdk.edge.power*), 87
 browse() (*cterasdk.core.portals.Portals* method), 62
 browse_global_admin() (*cterasdk.core.portals.Portals* method), 62

build() (*cterasdk.core.query.QueryParamBuilder* method), 63
 build() (*cterasdk.edge.query.QueryParamBuilder* method), 88
 by_name() (*cterasdk.core.devices.Devices* method), 56
 by_name() (*in module cterasdk.edge.taskmgr*), 96

C

C200 (*cterasdk.core.enum.DeviceType* attribute), 58
 C400 (*cterasdk.core.enum.DeviceType* attribute), 58
 C800 (*cterasdk.core.enum.DeviceType* attribute), 58
 C800P (*cterasdk.core.enum.DeviceType* attribute), 58
 Cache (*class in cterasdk.edge.cache*), 72
 caching (*cterasdk.edge.support.DebugLevel* attribute), 94
 CachingGateway (*cterasdk.edge.enum.OperationMode* attribute), 79
 CatalogReadOnlyMode (*cterasdk.edge.enum.SyncStatus* attribute), 81
 cbck (*cterasdk.edge.support.DebugLevel* attribute), 94
 CertificateServices (*class in cterasdk.client.ssl*), 48
 ChangeOwner (*cterasdk.edge.enum.AuditEvents* attribute), 76
 ChangePermissions (*cterasdk.edge.enum.AuditEvents* attribute), 76
 Checking (*cterasdk.edge.enum.VolumeStatus* attribute), 83
 CheckingQuota (*cterasdk.edge.enum.VolumeStatus* attribute), 83
 CIFSPacketSigning (*class in cterasdk.edge.enum*), 77
 CLASS (*cterasdk.convert.xml_types.XMLTypes* attribute), 49
 CLI (*class in cterasdk.edge.cli*), 73
 ClientSideCaching (*class in cterasdk.edge.enum*), 77
 ClockOutOfSync, 71
 ClocksOutOfSync (*cterasdk.edge.enum.BackupConfStatusID* attribute), 76
 ClocksOutOfSync (*cterasdk.edge.enum.SyncStatus* attribute), 81
 cloud_extender (*cterasdk.edge.support.DebugLevel* attribute), 94
 CloudBackup (*cterasdk.core.enum.LogTopic* attribute), 58
 CloudFS (*class in cterasdk.core.cloudfs*), 54
 CloudFSFolderFindingHelper (*class in cterasdk.core.types*), 65
 CloudPlug (*cterasdk.core.enum.DeviceType* attribute), 58
 CloudSync (*cterasdk.core.enum.LogTopic* attribute), 59

collaboration (*cterasdk.edge.support.DebugLevel attribute*), 94
 Command (*class in cterasdk.lib.cmd*), 100
 Completed (*cterasdk.edge.enum.TaskStatus attribute*), 82
 Completed (*cterasdk.edge.taskmgr.TaskStatusEnum attribute*), 96
 compute_zip_file_name () (*cterasdk.lib.filesystem.FileSystem static method*), 100
 Config (*class in cterasdk.edge.config*), 73
 configure () (*cterasdk.edge.backup.Backup method*), 71
 Configuring (*cterasdk.edge.enum.BackupConfStatusID attribute*), 76
 connect () (*cterasdk.edge.directoryservice.DirectoryService method*), 74
 connect () (*cterasdk.edge.services.Services method*), 89
 Connected (*cterasdk.edge.enum.ServicesConnectionState attribute*), 80
 Connected (*cterasdk.edge.enum.SyncStatus attribute*), 81
 connected () (*cterasdk.edge.services.Services method*), 89
 ConnectingFolders (*cterasdk.edge.enum.SyncStatus attribute*), 81
 ConnectionFailed (*cterasdk.edge.enum.SyncStatus attribute*), 81
 ConnectionTimeout, 106
 ConsentException, 106
 ContainsErrors (*cterasdk.edge.enum.VolumeStatus attribute*), 83
 ContentType (*class in cterasdk.client.http*), 47
 Context (*class in cterasdk.core.enum*), 57
 context (*cterasdk.object.Portal.GlobalAdmin attribute*), 104
 context (*cterasdk.object.Portal.Portal attribute*), 104
 context (*cterasdk.object.Portal.ServicesPortal attribute*), 105
 Converting (*cterasdk.edge.enum.VolumeStatus attribute*), 83
 copy () (*cterasdk.core.files.browser.FileBrowser method*), 50
 copy () (*in module cterasdk.core.files.cp*), 52
 copy_multi () (*cterasdk.core.files.browser.FileBrowser method*), 50
 copy_multi () (*in module cterasdk.core.files.cp*), 52
 Corrupted (*cterasdk.edge.enum.VolumeStatus attribute*), 83
 countLimit () (*cterasdk.core.query.QueryParamBuilder method*), 63
 countLimit () (*cterasdk.edge.query.QueryParamBuilder method*), 88
 CreateElement () (*in cterasdk.convert.format*), 49
 CreateFilesWriteData (*cterasdk.edge.enum.AuditEvents attribute*), 76
 CreateFoldersAppendData (*cterasdk.edge.enum.AuditEvents attribute*), 76
 CreateShareParam (*class in cterasdk.core.files.common*), 52
 CRITICAL (*cterasdk.core.enum.Severity attribute*), 60
 CRITICAL (*cterasdk.edge.enum.Severity attribute*), 80
 CTERAClient (*class in cterasdk.client.cteraclient*), 45
 CTERAClientException, 105
 CTERAConnectionError, 105
 CTERAException, 105
 CTERAHost (*class in cterasdk.client.host*), 46
 CTERAPath (*class in cterasdk.core.files.path*), 53
 CTERAPath (*class in cterasdk.edge.files.path*), 69
 cterasdk (*module*), 45
 cterasdk.client (*module*), 45
 cterasdk.client.cteraclient (*module*), 45
 cterasdk.client.host (*module*), 46
 cterasdk.client.http (*module*), 47
 cterasdk.client.ssl (*module*), 48
 cterasdk.common (*module*), 48
 cterasdk.common.item (*module*), 48
 cterasdk.common.object (*module*), 48
 cterasdk.config (*module*), 105
 cterasdk.convert (*module*), 48
 cterasdk.convert.exception (*module*), 49
 cterasdk.convert.format (*module*), 49
 cterasdk.convert.parse (*module*), 49
 cterasdk.convert.xml_types (*module*), 49
 cterasdk.core (*module*), 50
 cterasdk.core.activation (*module*), 53
 cterasdk.core.base_command (*module*), 54
 cterasdk.core.cloudfs (*module*), 54
 cterasdk.core.connection (*module*), 55
 cterasdk.core.decorator (*module*), 55
 cterasdk.core.devices (*module*), 55
 cterasdk.core.directoryservice (*module*), 57
 cterasdk.core.enum (*module*), 57
 cterasdk.core.files (*module*), 50
 cterasdk.core.files.browser (*module*), 50
 cterasdk.core.files.common (*module*), 52
 cterasdk.core.files.cp (*module*), 52
 cterasdk.core.files.directory (*module*), 52
 cterasdk.core.files.file_access (*module*), 52
 cterasdk.core.files.ln (*module*), 52
 cterasdk.core.files.ls (*module*), 53
 cterasdk.core.files.mv (*module*), 53
 cterasdk.core.files.path (*module*), 53

`cterasdk.core.files.recover (module)`, 53
`cterasdk.core.files.rename (module)`, 53
`cterasdk.core.files.rm (module)`, 53
`cterasdk.core.login (module)`, 61
`cterasdk.core.logs (module)`, 61
`cterasdk.core.portals (module)`, 61
`cterasdk.core.query (module)`, 63
`cterasdk.core.remote (module)`, 64
`cterasdk.core.reports (module)`, 62
`cterasdk.core.servers (module)`, 64
`cterasdk.core.session (module)`, 64
`cterasdk.core.types (module)`, 65
`cterasdk.core.union (module)`, 66
`cterasdk.core.users (module)`, 66
`cterasdk.core.zones (module)`, 67
`cterasdk.edge (module)`, 68
`cterasdk.edge.afp (module)`, 69
`cterasdk.edge.aio (module)`, 70
`cterasdk.edge.array (module)`, 70
`cterasdk.edge.audit (module)`, 70
`cterasdk.edge.backup (module)`, 71
`cterasdk.edge.base_command (module)`, 72
`cterasdk.edge.cache (module)`, 72
`cterasdk.edge.cli (module)`, 73
`cterasdk.edge.config (module)`, 73
`cterasdk.edge.connection (module)`, 74
`cterasdk.edge.decorator (module)`, 74
`cterasdk.edge.directoryservice (module)`, 74
`cterasdk.edge.drive (module)`, 75
`cterasdk.edge.enum (module)`, 75
`cterasdk.edge.files (module)`, 68
`cterasdk.edge.files.browser (module)`, 68
`cterasdk.edge.files.file_access (module)`, 69
`cterasdk.edge.files.mkdir (module)`, 69
`cterasdk.edge.files.path (module)`, 69
`cterasdk.edge.files.rm (module)`, 69
`cterasdk.edge.ftp (module)`, 83
`cterasdk.edge.groups (module)`, 84
`cterasdk.edge.licenses (module)`, 84
`cterasdk.edge.login (module)`, 85
`cterasdk.edge.logs (module)`, 85
`cterasdk.edge.mail (module)`, 85
`cterasdk.edge.network (module)`, 86
`cterasdk.edge.nfs (module)`, 86
`cterasdk.edge.ntp (module)`, 87
`cterasdk.edge.power (module)`, 87
`cterasdk.edge.query (module)`, 88
`cterasdk.edge.remote (module)`, 88
`cterasdk.edge.rsync (module)`, 88
`cterasdk.edge.services (module)`, 89
`cterasdk.edge.session (module)`, 90
`cterasdk.edge.shares (module)`, 91
`cterasdk.edge.shell (module)`, 93
`cterasdk.edge.smb (module)`, 93
`cterasdk.edge.support (module)`, 94
`cterasdk.edge.sync (module)`, 95
`cterasdk.edge.syslog (module)`, 95
`cterasdk.edge.taskmgr (module)`, 96
`cterasdk.edge.telnet (module)`, 96
`cterasdk.edge.timezone (module)`, 97
`cterasdk.edge.types (module)`, 97
`cterasdk.edge.uri (module)`, 98
`cterasdk.edge.users (module)`, 98
`cterasdk.edge.volumes (module)`, 99
`cterasdk.exception (module)`, 105
`cterasdk.lib (module)`, 100
`cterasdk.lib.cmd (module)`, 100
`cterasdk.lib.consent (module)`, 100
`cterasdk.lib.file_access_base (module)`, 100
`cterasdk.lib.filesystem (module)`, 100
`cterasdk.lib.iterator (module)`, 100
`cterasdk.lib.platform (module)`, 101
`cterasdk.lib.registry (module)`, 101
`cterasdk.lib.tempfile (module)`, 101
`cterasdk.lib.tracker (module)`, 101
`cterasdk.lib.version (module)`, 102
`cterasdk.object (module)`, 102
`cterasdk.object.Agent (module)`, 102
`cterasdk.object.Gateway (module)`, 102
`cterasdk.object.Portal (module)`, 104
`cterasdk.transcript (module)`, 105
`cterasdk.transcript.transcribe (module)`, 105
`cttp (cterasdk.edge.support.DebugLevel attribute)`, 94
`cttp_data (cterasdk.edge.support.DebugLevel attribute)`, 94

D

`db (cterasdk.edge.support.DebugLevel attribute)`, 94
`db () (cterasdk.client.cteraclient.CTERAClient method)`, 45
`db () (cterasdk.client.host.CTERAHost method)`, 46
`DEBUG (cterasdk.core.enum.Severity attribute)`, 60
`DEBUG (cterasdk.edge.enum.Severity attribute)`, 80
`debug (cterasdk.edge.support.DebugLevel attribute)`, 94
`DebugLevel (class in cterasdk.edge.support)`, 94
`default (cterasdk.core.cloudfs.CloudFS attribute)`, 54
`default (cterasdk.core.devices.Devices attribute)`, 56
`default (cterasdk.core.servers.Servers attribute)`, 64
`default_include (cterasdk.edge.logs.Logs attribute)`, 85
`defaultAuditEvents (cterasdk.edge.audit.Audit attribute)`, 70
`Delete (cterasdk.edge.enum.AuditEvents attribute)`, 76

`delete()` (*cterasdk.client.cteraclient.CTERAClient method*), 45
`delete()` (*cterasdk.client.host.CTERAHost method*), 46
`delete()` (*cterasdk.client.http.HTTPClient method*), 47
`delete()` (*cterasdk.core.cloudfs.CloudFS method*), 54
`delete()` (*cterasdk.core.files.browser.FileBrowser method*), 50
`delete()` (*cterasdk.core.portals.Portals method*), 62
`delete()` (*cterasdk.core.users.Users method*), 66
`delete()` (*cterasdk.core.zones.Zones method*), 67
`delete()` (*cterasdk.edge.array.Array method*), 70
`delete()` (*cterasdk.edge.files.browser.FileBrowser method*), 68
`delete()` (*cterasdk.edge.shares.Shares method*), 91
`delete()` (*cterasdk.edge.users.Users method*), 98
`delete()` (*cterasdk.edge.volumes.Volumes method*), 99
`delete()` (*in module cterasdk.core.files.rm*), 53
`delete()` (*in module cterasdk.edge.files.rm*), 69
`delete_all()` (*cterasdk.edge.array.Array method*), 70
`delete_all()` (*cterasdk.edge.volumes.Volumes method*), 99
`delete_multi()` (*cterasdk.core.files.browser.FileBrowser method*), 50
`delete_multi()` (*in module cterasdk.core.files.rm*), 53
DeleteSubfoldersAndFiles (*cterasdk.edge.enum.AuditEvents attribute*), 76
desktops() (*cterasdk.core.devices.Devices method*), 56
Device (*cterasdk.core.enum.OriginType attribute*), 59
device() (*cterasdk.core.devices.Devices method*), 56
Devices (*class in cterasdk.core.devices*), 55
devices() (*cterasdk.core.devices.Devices method*), 56
DeviceType (*class in cterasdk.core.enum*), 58
df() (*cterasdk.config.Logging static method*), 105
DG (*cterasdk.edge.enum.PrincipalType attribute*), 79
DirectoryService (*class in cterasdk.core.directoryservice*), 57
DirectoryService (*class in cterasdk.edge.directoryservice*), 74
disable() (*cterasdk.config.Logging static method*), 105
disable() (*cterasdk.edge.afp.AFP method*), 69
disable() (*cterasdk.edge.aio.AIO method*), 70
disable() (*cterasdk.edge.audit.Audit method*), 70
disable() (*cterasdk.edge.cache.Cache method*), 72
disable() (*cterasdk.edge.ftp.FTP method*), 83
disable() (*cterasdk.edge.mail.Mail method*), 85
disable() (*cterasdk.edge.nfs.NFS method*), 86
disable() (*cterasdk.edge.ntp.NTP method*), 87
disable() (*cterasdk.edge.rsync.RSync method*), 88
disable() (*cterasdk.edge.smb.SMB method*), 93
disable() (*cterasdk.edge.syslog.Syslog method*), 95
disable() (*cterasdk.edge.telnet.Telnet method*), 96
disable_abe() (*cterasdk.edge.smb.SMB method*), 93
disable_remote_access() (*cterasdk.edge.session.RemoteSession method*), 90
disable_sso() (*cterasdk.edge.services.Services method*), 89
disable_wizard() (*cterasdk.edge.config.Config method*), 73
Disabled (*cterasdk.core.enum.Role attribute*), 60
Disabled (*cterasdk.edge.enum.CIFSPacketSigning attribute*), 77
Disabled (*cterasdk.edge.enum.ClientSideCaching attribute*), 77
Disabled (*cterasdk.edge.enum.Mode attribute*), 78
Disabled (*cterasdk.edge.enum.OperationMode attribute*), 79
disconnect() (*cterasdk.edge.directoryservice.DirectoryService method*), 74
disconnect() (*cterasdk.edge.services.Services method*), 89
Disconnected (*cterasdk.edge.enum.ServicesConnectionState attribute*), 80
DisconnectedPortal (*cterasdk.edge.enum.SyncStatus attribute*), 81
dispatch() (*cterasdk.client.http.HttpClientBase method*), 47
dns (*cterasdk.edge.support.DebugLevel attribute*), 94
Documents (*cterasdk.edge.enum.ClientSideCaching attribute*), 77
domains() (*cterasdk.edge.directoryservice.DirectoryService method*), 74
download() (*cterasdk.client.cteraclient.CTERAClient method*), 45
download() (*cterasdk.core.files.browser.FileBrowser method*), 50
download() (*cterasdk.edge.files.browser.FileBrowser method*), 68
download() (*cterasdk.lib.file_access_base.FileAccessBase method*), 100
download_as_zip() (*cterasdk.core.files.browser.FileBrowser method*), 50
download_as_zip() (*cterasdk.edge.files.browser.FileBrowser method*), 68
download_as_zip() (*cterasdk.lib.file_access_base.FileAccessBase method*), 100
download_zip() (*cterasdk.client.cteraclient.CTERAClient method*), 45

`download_zip()` (*cterasdk.client.host.CTERAHost method*), 46
Drive (class in *cterasdk.edge.drive*), 75
DU (*cterasdk.edge.enum.PrincipalType* attribute), 79

E

EMERGENCY (*cterasdk.core.enum.Severity* attribute), 60
EMERGENCY (*cterasdk.edge.enum.Severity* attribute), 80
`enable()` (*cterasdk.config.Logging* static method), 105
`enable()` (*cterasdk.edge.aio.AIO* method), 70
`enable()` (*cterasdk.edge.audit.Audit* method), 71
`enable()` (*cterasdk.edge.cache.Cache* method), 72
`enable()` (*cterasdk.edge.ftp.FTP* method), 83
`enable()` (*cterasdk.edge.mail.Mail* method), 85
`enable()` (*cterasdk.edge.nfs.NFS* method), 86
`enable()` (*cterasdk.edge.ntp.NTP* method), 87
`enable()` (*cterasdk.edge.rsync.RSync* method), 88
`enable()` (*cterasdk.edge.smb.SMB* method), 93
`enable()` (*cterasdk.edge.syslog.Syslog* method), 95
`enable()` (*cterasdk.edge.telnet.Telnet* method), 96
`enable_abe()` (*cterasdk.edge.smb.SMB* method), 93
`enable_dhcp()` (*cterasdk.edge.network.Network* method), 86
`enable_remote_access()` (*cterasdk.edge.session.RemoteSession* method), 90
`enable_sso()` (*cterasdk.edge.services.Services* method), 89
`enable_wizard()` (*cterasdk.edge.config.Config* method), 73
Enabled (*cterasdk.edge.enum.Mode* attribute), 78
`encoded_fullpath()` (*cterasdk.core.files.path.CTERAPath* method), 53
`encoded_fullpath()` (*cterasdk.edge.files.path.CTERAPath* method), 69
`encoded_parent()` (*cterasdk.core.files.path.CTERAPath* method), 53
`encoded_parent()` (*cterasdk.edge.files.path.CTERAPath* method), 69
EncryptionMode (class in *cterasdk.edge.backup*), 71
EndUser (*cterasdk.core.enum.Role* attribute), 60
`eq()` (*cterasdk.core.query.FilterBuilder* method), 63
EQUALS (*cterasdk.core.query.Restriction* attribute), 63
ERROR (*cterasdk.core.enum.Severity* attribute), 60
ERROR (*cterasdk.edge.enum.Severity* attribute), 80
`error()` (*cterasdk.edge.support.DebugLevel* attribute), 94
`error_abort()` (*cterasdk.edge.support.DebugLevel* attribute), 94
ErrorStatus, 101
EV128 (*cterasdk.edge.enum.License* attribute), 78
EV16 (*cterasdk.edge.enum.License* attribute), 78
EV32 (*cterasdk.edge.enum.License* attribute), 78

EV4 (*cterasdk.edge.enum.License* attribute), 78
EV64 (*cterasdk.edge.enum.License* attribute), 78
EV8 (*cterasdk.edge.enum.License* attribute), 78
Everyone (*cterasdk.edge.enum.LocalGroup* attribute), 78
`evictor()` (*cterasdk.edge.support.DebugLevel* attribute), 94
`evictor_verbose()` (*cterasdk.edge.support.DebugLevel* attribute), 94
`execute()` (*cterasdk.client.cteraclient.CTERAClient* method), 45
`execute()` (*cterasdk.client.host.CTERAHost* method), 46
ExhaustedException, 106
`exists()` (*cterasdk.lib.filesystem.FileSystem* static method), 100
`expanduser()` (*cterasdk.lib.filesystem.FileSystem* static method), 100

F

Failed (*cterasdk.edge.enum.BackupConfStatusID* attribute), 76
Failed (*cterasdk.edge.enum.TaskStatus* attribute), 82
Failed (*cterasdk.edge.taskmgr.TaskStatusEnum* attribute), 96
`failed()` (*cterasdk.lib.tracker.StatusTracker* method), 101
FailedFilesInReadOnlyFolder (*cterasdk.edge.enum.SyncStatus* attribute), 81
`fetch()` (*cterasdk.core.directoryservice.DirectoryService* method), 57
`file_browser_base_path` (*cterasdk.object.Portal.GlobalAdmin* attribute), 104
`file_browser_base_path` (*cterasdk.object.Portal.Portal* attribute), 104
`file_browser_base_path` (*cterasdk.object.Portal.ServicesPortal* attribute), 105
`file_descriptor()` (*cterasdk.client.cteraclient.CTERAClient* static method), 45
FileAccess (class in *cterasdk.core.files.file_access*), 52
FileAccess (class in *cterasdk.edge.files.file_access*), 69
FileAccessBase (class in *cterasdk.lib.file_access_base*), 100
FileAccessMode (class in *cterasdk.edge.enum*), 77
FileBrowser (class in *cterasdk.core.files.browser*), 50
FileBrowser (class in *cterasdk.edge.files.browser*), 68
`filers()` (*cterasdk.core.devices.Devices* method), 56

files (*cterasdk.edge.support.DebugLevel attribute*), 94
 files () (*in module cterasdk.edge.uri*), 98
 FileSystem (*class in cterasdk.lib.filesystem*), 100
 FileSystemException, 106
 Filter (*class in cterasdk.core.query*), 63
 FilterBuilder (*class in cterasdk.core.query*), 63
 FilterType (*class in cterasdk.core.query*), 63
 find () (*cterasdk.core.cloudfs.CloudFS method*), 54
 fmt () (*cterasdk.config.Logging static method*), 105
 folder_groups () (*cterasdk.core.reports.Reports method*), 62
 folders () (*cterasdk.core.reports.Reports method*), 62
 Forbidden, 69
 force_eviction () (*cterasdk.edge.cache.Cache method*), 72
 form_data () (*cterasdk.client.cteraclient.CTERAClient method*), 45
 form_data () (*cterasdk.client.host.CTERAHost method*), 46
 format () (*cterasdk.edge.drive.Drive method*), 75
 format_all () (*cterasdk.edge.drive.Drive method*), 75
 Formatting (*cterasdk.edge.enum.VolumeStatus attribute*), 83
 from_server_object () (*cterasdk.edge.types.ShareAccessControlEntry static method*), 97
 from_server_object () (*cterasdk.edge.types.UserGroupEntry static method*), 98
 fromjsonstr () (*in module cterasdk.convert.parse*), 49
 fromValue () (*cterasdk.core.query.FilterType static method*), 63
 fromxmlstr () (*cterasdk.client.cteraclient.CTERAClient static method*), 45
 fromxmlstr () (*in module cterasdk.convert.parse*), 49
 FTP (*class in cterasdk.edge.ftp*), 83
 fullpath () (*cterasdk.core.files.path.CTERAPath method*), 53
 fullpath () (*cterasdk.edge.files.path.CTERAPath method*), 69

G

Gateway (*class in cterasdk.object.Gateway*), 102
 Gateways (*cterasdk.core.enum.DeviceType attribute*), 58
 ge () (*cterasdk.core.query.FilterBuilder method*), 63
 generate_code () (*cterasdk.core.activation.Activation method*), 54
 get () (*cterasdk.client.cteraclient.CTERAClient method*), 45
 get () (*cterasdk.client.host.CTERAHost method*), 46
 get () (*cterasdk.client.http.HTTPClient method*), 47
 get () (*cterasdk.config.Logging static method*), 105
 get () (*cterasdk.core.logs.Logs method*), 61
 get () (*cterasdk.core.users.Users method*), 66
 get () (*cterasdk.core.zones.Zones method*), 67
 get () (*cterasdk.edge.array.Array method*), 70
 get () (*cterasdk.edge.drive.Drive method*), 75
 get () (*cterasdk.edge.groups.Groups method*), 84
 get () (*cterasdk.edge.licenses.Licenses method*), 84
 get () (*cterasdk.edge.shares.Shares method*), 91
 get () (*cterasdk.edge.users.Users method*), 98
 get () (*cterasdk.edge.volumes.Volumes method*), 99
 get () (*cterasdk.lib.registry.Registry method*), 101
 get_configuration () (*cterasdk.edge.ftp.FTP method*), 83
 get_configuration () (*cterasdk.edge.nfs.NFS method*), 87
 get_configuration () (*cterasdk.edge.rsync.RSync method*), 88
 get_configuration () (*cterasdk.edge.smb.SMB method*), 93
 get_configuration () (*cterasdk.edge.syslog.Syslog method*), 96
 get_connected_domain () (*cterasdk.edge.directoryservice.DirectoryService method*), 74
 get_dirpath () (*cterasdk.lib.filesystem.FileSystem method*), 100
 get_hostname () (*cterasdk.edge.config.Config method*), 73
 get_local_file_info () (*cterasdk.lib.filesystem.FileSystem static method*), 100
 get_location () (*cterasdk.edge.config.Config method*), 73
 get_multi () (*cterasdk.client.cteraclient.CTERAClient method*), 45
 get_multi () (*cterasdk.client.host.CTERAHost method*), 46
 get_status () (*cterasdk.edge.drive.Drive method*), 75
 get_status () (*cterasdk.edge.network.Network method*), 86
 get_status () (*cterasdk.edge.services.Services method*), 89
 get_status () (*cterasdk.edge.sync.Sync method*), 95
 get_support_report () (*cterasdk.edge.support.Support method*), 95
 get_timezone () (*cterasdk.edge.timezone.Timezone method*), 97
 getcode () (*cterasdk.client.http.HTTPResponse method*), 47
 GetFoldersList (*cterasdk.edge.enum.BackupConfStatusID attribute*), 76
 geturi () (*in module cterasdk.client.http*), 48
 geturl () (*cterasdk.client.http.HTTPResponse*)

method), 47
 global_admin() (cterasdk.core.session.Session method), 64
 GlobalAdmin (class in cterasdk.object.Portal), 104
 GREATER_EQUALS (cterasdk.core.query.Restriction attribute), 64
 GREATER_THAN (cterasdk.core.query.Restriction attribute), 64
 Group (cterasdk.core.enum.PortalAccountType attribute), 59
 GroupAccount (class in cterasdk.core.types), 65
 Groups (class in cterasdk.edge.groups), 84
 Groups (cterasdk.core.enum.SearchType attribute), 60
 gt() (cterasdk.core.query.FilterBuilder method), 63

H

host() (cterasdk.client.host.NetworkHost method), 47
 HostUnreachable, 106
 http (cterasdk.edge.support.DebugLevel attribute), 94
 HTTPClient (class in cterasdk.client.http), 47
 HttpClientBase (class in cterasdk.client.http), 47
 HttpClientRequest (class in cterasdk.client.http), 48
 HttpClientRequestDelete (class in cterasdk.client.http), 48
 HttpClientRequestGet (class in cterasdk.client.http), 48
 HttpClientRequestMkcol (class in cterasdk.client.http), 48
 HttpClientRequestPost (class in cterasdk.client.http), 48
 HttpClientRequestPut (class in cterasdk.client.http), 48
 HTTPException, 47
 HTTPResponse (class in cterasdk.client.http), 47
 https() (cterasdk.client.host.NetworkHost method), 47

I

ID (cterasdk.convert.xml_types.XMLTypes attribute), 49
 IfClientAgrees (cterasdk.edge.enum.CIFSPacketSigning attribute), 77
 ifconfig() (cterasdk.edge.network.Network method), 86
 Inactive (cterasdk.core.session.SessionStatus attribute), 65
 Inactive (cterasdk.edge.session.SessionStatus attribute), 90
 inactive_session() (in module cterasdk.core.session), 65
 inactive_session() (in module cterasdk.edge.session), 90
 InactiveSession (class in cterasdk.edge.session), 90

include() (cterasdk.core.query.QueryParamBuilder method), 63
 include() (cterasdk.edge.query.QueryParamBuilder method), 88
 include_classname() (cterasdk.core.query.QueryParamBuilder method), 63
 include_classname() (cterasdk.core.query.QueryParams method), 63
 include_classname() (cterasdk.edge.query.QueryParam method), 88
 include_classname() (cterasdk.edge.query.QueryParamBuilder method), 88
 IncorrectPassphrase, 72
 increment() (cterasdk.core.query.QueryParams method), 63
 increment() (cterasdk.edge.query.QueryParam method), 88
 increment() (cterasdk.edge.taskmgr.Task method), 96
 increment() (cterasdk.lib.tracker.StatusTracker method), 101
 index (cterasdk.edge.support.DebugLevel attribute), 94
 infer() (cterasdk.edge.licenses.Licenses static method), 84
 INFO (cterasdk.core.enum.Severity attribute), 60
 INFO (cterasdk.edge.enum.Severity attribute), 80
 info (cterasdk.edge.support.DebugLevel attribute), 94
 initialize() (cterasdk.core.session.Session method), 64
 Initializing (cterasdk.core.session.SessionStatus attribute), 65
 initializing() (cterasdk.core.session.Session method), 64
 InitializingConnection (cterasdk.edge.enum.SyncStatus attribute), 81
 InputError, 106
 instance() (cterasdk.core.files.common.ActionResourcesParam static method), 52
 instance() (cterasdk.core.files.common.CreateShareParam static method), 52
 instance() (cterasdk.core.files.common.SrcDstParam static method), 52
 instance() (cterasdk.lib.filesystem.FileSystem static method), 100
 instance() (cterasdk.lib.platform.Platform static method), 101
 instance() (cterasdk.lib.registry.Registry static method), 101
 instance() (cterasdk.lib.version.Version static method), 102
 InternalError (cterasdk.edge.enum.SyncStatus attribute), 81

InvalidAverageBlockSize
(*cterasdk.edge.enum.SyncStatus* attribute), 81

InvalidConfiguration
(*cterasdk.edge.enum.SyncStatus* attribute), 81

InvalidName, 52

InvalidPath, 52

ipconfig() (*cterasdk.edge.network.Network* method), 86

IPProtocol (class in *cterasdk.edge.enum*), 77

is_configured() (*cterasdk.edge.backup.Backup* method), 71

is_disabled() (*cterasdk.edge.afp.AFP* method), 69

is_disabled() (*cterasdk.edge.ftp.FTP* method), 83

is_disabled() (*cterasdk.edge.nfs.NFS* method), 87

is_disabled() (*cterasdk.edge.rsync.RSync* method), 88

is_disabled() (*cterasdk.edge.sync.Sync* method), 95

is_enabled() (*cterasdk.edge.aio.AIO* method), 70

is_enabled() (*cterasdk.edge.sync.Sync* method), 95

is_local (*cterasdk.core.types.PortalAccount* attribute), 65

is_nosession() (in module *cterasdk.edge.decorator*), 74

is_wizard_enabled() (*cterasdk.edge.config.Config* method), 73

Item (class in *cterasdk.common.item*), 48

ItemExists, 52, 69

Iterator (class in *cterasdk.lib.iterator*), 100

iterator() (*cterasdk.object.Portal.Portal* method), 104

iterator() (in module *cterasdk.core.query*), 64

J

JBOD (*cterasdk.edge.enum.RAIDLevel* attribute), 79

join() (*cterasdk.exception.CTERAException* method), 105

joinpath() (*cterasdk.core.files.path.CTERAPath* method), 53

joinpath() (*cterasdk.edge.files.path.CTERAPath* method), 69

K

KeyRequired (*cterasdk.edge.enum.VolumeStatus* attribute), 83

L

le() (*cterasdk.core.query.FilterBuilder* method), 63

LESS_EQUALS (*cterasdk.core.query.Restriction* attribute), 64

LESS_THAN (*cterasdk.core.query.Restriction* attribute), 64

LG (*cterasdk.edge.enum.PrincipalType* attribute), 79

License (class in *cterasdk.edge.enum*), 78

license (*cterasdk.edge.support.DebugLevel* attribute), 94

Licenses (class in *cterasdk.edge.licenses*), 84

LIKE (*cterasdk.core.query.Restriction* attribute), 64

like() (*cterasdk.core.query.FilterBuilder* method), 63

LIST (*cterasdk.convert.xml_types.XMLTypes* attribute), 49

list_dir() (in module *cterasdk.core.files.ls*), 53

list_domain_users() (*cterasdk.core.users.Users* method), 66

list_domains() (*cterasdk.core.users.Users* method), 66

list_folder_groups() (*cterasdk.core.cloudfs.CloudFS* method), 54

list_folders() (*cterasdk.core.cloudfs.CloudFS* method), 54

list_local_users() (*cterasdk.core.users.Users* method), 67

list_servers() (*cterasdk.core.servers.Servers* method), 64

ListFolderReadData (*cterasdk.edge.enum.AuditEvents* attribute), 76

Local (*cterasdk.edge.session.SessionType* attribute), 90

local() (*cterasdk.edge.session.Session* method), 90

local() (in module *cterasdk.edge.uri*), 98

LocalDirectoryNotFound, 106

LocalFileNotFound, 106

LocalGroup (class in *cterasdk.edge.enum*), 78

LocalSession (class in *cterasdk.edge.session*), 90

Logging (class in *cterasdk.config*), 105

Login (class in *cterasdk.core.login*), 61

Login (class in *cterasdk.edge.login*), 85

login() (*cterasdk.client.host.CTERAHost* method), 46

login() (*cterasdk.core.login.Login* method), 61

login() (*cterasdk.edge.login.Login* method), 85

login() (in module *cterasdk.edge.remote*), 88

logout() (*cterasdk.client.host.CTERAHost* method), 46

logout() (*cterasdk.core.login.Login* method), 61

logout() (*cterasdk.edge.login.Login* method), 85

Logs (class in *cterasdk.core.logs*), 61

Logs (class in *cterasdk.edge.logs*), 85

logs() (*cterasdk.edge.logs.Logs* method), 85

LogTopic (class in *cterasdk.core.enum*), 58

ls() (*cterasdk.core.files.browser.FileBrowser* method), 51

ls() (*cterasdk.edge.files.browser.FileBrowser* static method), 68

ls() (in module *cterasdk.core.files.ls*), 53

lt() (*cterasdk.core.query.FilterBuilder* method), 63

LU (*cterasdk.edge.enum.PrincipalType* attribute), 79

M

Mail (class in *cterasdk.edge.mail*), 85

make_local_files_dir() (cterasdk.object.Gateway.Gateway static method), 103

Manual (cterasdk.edge.enum.ClientSideCaching attribute), 77

mkcol() (cterasdk.client.cteraclient.CTERAClient method), 45

mkcol() (cterasdk.client.host.CTERAHost method), 46

mkcol() (cterasdk.client.http.HTTPClient method), 47

mkdir() (cterasdk.core.cloudfs.CloudFS method), 54

mkdir() (cterasdk.core.files.browser.FileBrowser method), 51

mkdir() (cterasdk.edge.files.browser.FileBrowser method), 68

mkdir() (cterasdk.lib.tempfile.TempfileServices static method), 101

mkdir() (in module *cterasdk.core.files.directory*), 52

mkdir() (in module *cterasdk.edge.files.mkdir*), 69

mkfg() (cterasdk.core.cloudfs.CloudFS method), 55

mkfile() (cterasdk.lib.tempfile.TempfileServices static method), 101

mlink() (cterasdk.core.files.browser.FileBrowser method), 51

mlink() (in module *cterasdk.core.files.ln*), 52

mkpath() (cterasdk.core.files.browser.FileBrowser method), 51

mkpath() (cterasdk.edge.files.browser.FileBrowser static method), 68

Mode (class in *cterasdk.edge.enum*), 78

modify() (cterasdk.edge.ftp.FTP method), 83

modify() (cterasdk.edge.nfs.NFS method), 87

modify() (cterasdk.edge.rsync.RSync method), 88

modify() (cterasdk.edge.shares.Shares method), 92

modify() (cterasdk.edge.smb.SMB method), 93

modify() (cterasdk.edge.syslog.Syslog method), 96

modify() (cterasdk.edge.users.Users method), 99

modify() (cterasdk.edge.volumes.Volumes method), 99

Mounting (cterasdk.edge.enum.VolumeStatus attribute), 83

move() (cterasdk.core.files.browser.FileBrowser method), 51

move() (in module *cterasdk.core.files.mv*), 53

move_multi() (cterasdk.core.files.browser.FileBrowser method), 51

move_multi() (in module *cterasdk.core.files.mv*), 53

N

NA (cterasdk.edge.enum.FileAccessMode attribute), 77

name (cterasdk.core.types.CloudFSFolderFindingHelper attribute), 65

name (cterasdk.edge.types.ShareAccessControlEntry attribute), 97

name() (cterasdk.core.files.path.CTERAPath method), 53

name() (cterasdk.edge.files.path.CTERAPath method), 69

name_attr (cterasdk.core.devices.Devices attribute), 57

ne() (cterasdk.core.query.FilterBuilder method), 63

Network (class in *cterasdk.edge.network*), 86

NetworkHost (class in *cterasdk.client.host*), 47

NFS (class in *cterasdk.edge.nfs*), 86

NoFolder (cterasdk.edge.enum.BackupConfStatusID attribute), 76

NoFolder (cterasdk.edge.enum.SyncStatus attribute), 81

NONE (cterasdk.core.enum.PolicyType attribute), 59

none (cterasdk.edge.support.DebugLevel attribute), 94

NOT_EQUALS (cterasdk.core.query.Restriction attribute), 64

NotFound, 72

NOTICE (cterasdk.core.enum.Severity attribute), 60

NOTICE (cterasdk.edge.enum.Severity attribute), 80

NotInitialized (cterasdk.edge.enum.BackupConfStatusID attribute), 76

NotInitialized (cterasdk.edge.enum.SyncStatus attribute), 81

notLike() (cterasdk.core.query.FilterBuilder method), 63

NTP (class in *cterasdk.edge.ntp*), 87

ntp (cterasdk.edge.support.DebugLevel attribute), 94

O

OBJ (cterasdk.convert.xml_types.XMLTypes attribute), 50

Object (class in *cterasdk.common.object*), 48

obtain_tenant() (in module *cterasdk.core.session*), 65

obtain_ticket() (in module *cterasdk.edge.remote*), 88

obtain_user() (in module *cterasdk.core.session*), 65

Off (cterasdk.edge.enum.SyncStatus attribute), 81

Ok (cterasdk.edge.enum.VolumeStatus attribute), 83

on_ssl_error() (cterasdk.client.http.HttpClientBase method), 47

on_timeout() (cterasdk.client.http.HttpClientBase static method), 47

OnlyAuthenticatedUsers (cterasdk.edge.enum.Acl attribute), 75

Open (cterasdk.edge.enum.TCPConnectRC attribute), 82

openfile() (cterasdk.client.host.CTERAHost method), 46

OperationMode (class in *cterasdk.edge.enum*), 79

orFilter() (cterasdk.core.query.QueryParamBuilder method), 63

OriginType (class in *cterasdk.core.enum*), 59

`os()` (*cterasdk.lib.platform.Platform* method), 101
`OutOfQuota` (*cterasdk.edge.enum.SyncStatus* attribute), 81
`owner` (*cterasdk.core.types.CloudFSFolderFindingHelper* attribute), 65

P

`parent()` (*cterasdk.core.files.path.CTERAPath* method), 53
`parent()` (*cterasdk.edge.files.path.CTERAPath* method), 69
`ParseException`, 49
`ParserException`, 106
`ParseValue()` (in module *cterasdk.convert.parse*), 49
`parts()` (*cterasdk.core.files.path.CTERAPath* method), 53
`parts()` (*cterasdk.edge.files.path.CTERAPath* method), 69
`perm` (*cterasdk.edge.types.ShareAccessControlEntry* attribute), 97
`pin()` (*cterasdk.edge.cache.Cache* method), 72
`pin_all()` (*cterasdk.edge.cache.Cache* method), 72
`pin_exclude()` (*cterasdk.edge.cache.Cache* method), 72
`Platform` (class in *cterasdk.lib.platform*), 101
`PolicyType` (class in *cterasdk.core.enum*), 59
`port()` (*cterasdk.client.host.NetworkHost* method), 47
`Portal` (class in *cterasdk.object.Portal*), 104
`Portal` (*cterasdk.core.enum.OriginType* attribute), 59
`PortalAccount` (class in *cterasdk.core.types*), 65
`PortalAccountType` (class in *cterasdk.core.enum*), 59
`Portals` (class in *cterasdk.core.portals*), 61
`portals()` (*cterasdk.core.reports.Reports* method), 62
`post()` (*cterasdk.client.cteraclient.CTERAClient* method), 45
`post()` (*cterasdk.client.host.CTERAHost* method), 46
`post()` (*cterasdk.client.http.HTTPClient* method), 47
`Power` (class in *cterasdk.edge.power*), 87
`prettify()` (in module *cterasdk.transcript.transcribe*), 105
`principal_type` (*cterasdk.edge.types.ShareAccessControlEntry* attribute), 97
`principal_type` (*cterasdk.edge.types.UserGroupEntry* attribute), 98
`PrincipalType` (class in *cterasdk.edge.enum*), 79
`process` (*cterasdk.edge.support.DebugLevel* attribute), 94
`put()` (*cterasdk.client.cteraclient.CTERAClient* method), 45
`put()` (*cterasdk.client.host.CTERAHost* method), 46
`put()` (*cterasdk.client.http.HTTPClient* method), 47
`put()` (*cterasdk.core.query.QueryParamBuilder* method), 63

`put()` (*cterasdk.edge.query.QueryParamBuilder* method), 88
`put()` (*cterasdk.exception.CTERAException* method), 105
`put()` (*cterasdk.object.Portal.Portal* method), 104
`python_version()` (*cterasdk.lib.platform.Platform* method), 101
`PythonVersionException`, 106

Q

`query()` (*cterasdk.object.Gateway.Gateway* method), 103
`query()` (*cterasdk.object.Portal.Portal* method), 104
`query()` (in module *cterasdk.core.query*), 64
`query()` (in module *cterasdk.edge.query*), 88
`QueryParam` (class in *cterasdk.edge.query*), 88
`QueryParamBuilder` (class in *cterasdk.core.query*), 63
`QueryParamBuilder` (class in *cterasdk.edge.query*), 88
`QueryParams` (class in *cterasdk.core.query*), 63

R

`RAID_0` (*cterasdk.edge.enum.RAIDLevel* attribute), 79
`RAID_1` (*cterasdk.edge.enum.RAIDLevel* attribute), 79
`RAID_5` (*cterasdk.edge.enum.RAIDLevel* attribute), 79
`RAID_6` (*cterasdk.edge.enum.RAIDLevel* attribute), 79
`RAIDLevel` (class in *cterasdk.edge.enum*), 79
`read()` (*cterasdk.client.http.HTTPResponse* method), 47
`ReadExtendedAttributes` (*cterasdk.edge.enum.AuditEvents* attribute), 76
`ReadOnly` (*cterasdk.edge.enum.VolumeStatus* attribute), 83
`ReadOnlyAdmin` (*cterasdk.core.enum.Role* attribute), 60
`ReadOnlyAdministrators` (*cterasdk.edge.enum.LocalGroup* attribute), 78
`ReadWriteAdmin` (*cterasdk.core.enum.Role* attribute), 60
`reboot()` (*cterasdk.edge.power.Power* method), 87
`reconnect()` (*cterasdk.edge.services.Services* method), 89
`Recoverable` (*cterasdk.edge.backup.EncryptionMode* attribute), 72
`Recovering` (*cterasdk.edge.enum.VolumeStatus* attribute), 83
`refresh()` (*cterasdk.edge.sync.Sync* method), 95
`register()` (*cterasdk.lib.registry.Registry* method), 101
`register_session()` (*cterasdk.client.host.CTERAHost* method), 46
`Registry` (class in *cterasdk.lib.registry*), 101

- RejectedByPolicy (*cterasdk.edge.enum.SyncStatus attribute*), 81
- Remote (*cterasdk.edge.session.SessionType attribute*), 90
- remote() (*cterasdk.edge.session.Session method*), 90
- remote() (*in module cterasdk.edge.uri*), 98
- remote_access() (*cterasdk.edge.session.RemoteSession method*), 90
- remote_access() (*cterasdk.object.Gateway.Gateway method*), 103
- remote_access() (*in module cterasdk.edge.remote*), 88
- remote_access() (*in module cterasdk.edge.uri*), 98
- remote_command() (*in module cterasdk.core.remote*), 64
- remote_from() (*cterasdk.edge.session.RemoteSession method*), 90
- RemoteDirectoryNotFound, 106
- RemoteFileSystemException, 106
- RemoteSession (*class in cterasdk.edge.session*), 90
- remove() (*cterasdk.lib.registry.Registry method*), 101
- remove_acl() (*cterasdk.edge.shares.Shares method*), 92
- remove_members() (*cterasdk.edge.groups.Groups method*), 84
- remove_pin() (*cterasdk.edge.cache.Cache method*), 72
- RemoveShareAccessControlEntry (*class in cterasdk.edge.types*), 97
- rename() (*cterasdk.core.files.browser.FileBrowser method*), 51
- rename() (*cterasdk.lib.filesystem.FileSystem method*), 100
- rename() (*in module cterasdk.core.files.rename*), 53
- RenameException, 106
- Repairing (*cterasdk.edge.enum.VolumeStatus attribute*), 83
- Reports (*class in cterasdk.core.reports*), 62
- Required (*cterasdk.edge.enum.CIFS PacketSigning attribute*), 77
- ReservedName, 52
- reset() (*cterasdk.edge.power.Power method*), 87
- Resizing (*cterasdk.edge.enum.VolumeStatus attribute*), 83
- resolve() (*cterasdk.edge.taskmgr.Task static method*), 96
- resolve() (*cterasdk.lib.tracker.StatusTracker method*), 101
- Restriction (*class in cterasdk.core.query*), 63
- rm() (*cterasdk.object.Gateway.Gateway method*), 103
- rmdir() (*cterasdk.lib.tempfile.TempfileServices static method*), 101
- rmfg() (*cterasdk.core.cloudfs.CloudFS method*), 55
- RO (*cterasdk.edge.enum.FileAccessMode attribute*), 77
- Role (*class in cterasdk.core.enum*), 59
- RSync (*class in cterasdk.edge.rsycn*), 88
- rsync (*cterasdk.edge.support.DebugLevel attribute*), 94
- run_command() (*cterasdk.edge.cli.CLI method*), 73
- run_command() (*cterasdk.edge.shell.Shell method*), 93
- Running (*cterasdk.edge.enum.TaskStatus attribute*), 82
- Running (*cterasdk.edge.taskmgr.TaskStatusEnum attribute*), 96
- running() (*cterasdk.lib.tracker.StatusTracker method*), 101
- running() (*in module cterasdk.edge.taskmgr*), 96
- RW (*cterasdk.edge.enum.FileAccessMode attribute*), 77
- ## S
- samba (*cterasdk.edge.support.DebugLevel attribute*), 94
- save() (*cterasdk.lib.filesystem.FileSystem method*), 100
- save_cert_from_server() (*cterasdk.client.ssl.CertificateServices static method*), 48
- Scanning (*cterasdk.edge.enum.SyncStatus attribute*), 81
- scheme() (*cterasdk.client.host.NetworkHost method*), 47
- SearchType (*class in cterasdk.core.enum*), 60
- Secret (*cterasdk.edge.backup.EncryptionMode attribute*), 72
- SELECT (*cterasdk.core.enum.PolicyType attribute*), 59
- ServerAgent (*cterasdk.core.enum.DeviceType attribute*), 58
- Servers (*class in cterasdk.core.servers*), 64
- servers() (*cterasdk.core.devices.Devices method*), 57
- Services (*class in cterasdk.edge.services*), 89
- ServicesConnectionState (*class in cterasdk.edge.enum*), 79
- ServicesPortal (*class in cterasdk.object.Portal*), 105
- ServicesPortal (*cterasdk.core.enum.Context attribute*), 57
- ServiceUnavailable (*cterasdk.edge.enum.SyncStatus attribute*), 82
- Session (*class in cterasdk.core.session*), 64
- Session (*class in cterasdk.edge.session*), 90
- session() (*cterasdk.client.host.CTERAHost method*), 46
- session() (*cterasdk.core.base_command.BaseCommand method*), 54
- session() (*cterasdk.edge.base_command.BaseCommand method*), 72
- SessionStatus (*class in cterasdk.core.session*), 65
- SessionStatus (*class in cterasdk.edge.session*), 90
- SessionType (*class in cterasdk.edge.session*), 90
- set_acl() (*cterasdk.edge.shares.Shares method*), 92

`set_debug_level()` (*cterasdk.edge.support.Support method*), 95
`set_hostname()` (*cterasdk.edge.config.Config method*), 73
`set_location()` (*cterasdk.edge.config.Config method*), 73
`set_packet_signing()` (*cterasdk.edge.smb.SMB method*), 93
`set_share_winaccls()` (*cterasdk.edge.shares.Shares method*), 92
`set_static_ipaddr()` (*cterasdk.edge.network.Network method*), 86
`set_static_nameserver()` (*cterasdk.edge.network.Network method*), 86
`set_timezone()` (*cterasdk.edge.timezone.Timezone method*), 97
`SetAppendValue()` (in module *cterasdk.convert.parse*), 49
`setLevel()` (*cterasdk.config.Logging static method*), 105
`setValue()` (*cterasdk.core.query.FilterBuilder method*), 63
`Severity` (class in *cterasdk.core.enum*), 60
`Severity` (class in *cterasdk.edge.enum*), 80
`ShareAccessControlEntry` (class in *cterasdk.edge.types*), 97
`Shares` (class in *cterasdk.edge.shares*), 91
`Shell` (class in *cterasdk.edge.shell*), 93
`should_trust()` (*cterasdk.client.http.HttpClientBase method*), 47
`ShouldSupportWinNtAcl` (*cterasdk.edge.enum.SyncStatus attribute*), 82
`show()` (*cterasdk.client.host.CTERAHost method*), 46
`show()` (in module *cterasdk.core.query*), 64
`show()` (in module *cterasdk.edge.query*), 88
`show_multi()` (*cterasdk.client.host.CTERAHost method*), 46
`show_query()` (*cterasdk.object.Gateway.Gateway method*), 103
`show_query()` (*cterasdk.object.Portals.Portals method*), 104
`shutdown()` (*cterasdk.edge.power.Power method*), 87
`SMB` (class in *cterasdk.edge.smb*), 93
`sortBy()` (*cterasdk.core.query.QueryParamBuilder method*), 63
`sortBy()` (*cterasdk.edge.query.QueryParamBuilder method*), 88
`SrcDstParam` (class in *cterasdk.core.files.common*), 52
`SSLException`, 106
`sso_enabled()` (*cterasdk.edge.services.Services method*), 89
`start()` (*cterasdk.edge.backup.Backup method*), 71
`start_local_session()` (in module *cterasdk.edge.session*), 90
`start_remote_session()` (in module *cterasdk.edge.session*), 90
`startFrom()` (*cterasdk.core.query.QueryParamBuilder method*), 63
`startFrom()` (*cterasdk.edge.query.QueryParamBuilder method*), 88
`StatusTracker` (class in *cterasdk.lib.tracker*), 101
`storage` (*cterasdk.edge.support.DebugLevel attribute*), 94
`storage()` (*cterasdk.core.reports.Reports method*), 62
`successful()` (*cterasdk.lib.tracker.StatusTracker method*), 101
`Support` (class in *cterasdk.edge.support*), 95
`Support` (*cterasdk.core.enum.Role attribute*), 60
`suspend()` (*cterasdk.edge.backup.Backup method*), 71
`suspend()` (*cterasdk.edge.sync.Sync method*), 95
`Sync` (class in *cterasdk.edge.sync*), 95
`Synced` (*cterasdk.edge.enum.SyncStatus attribute*), 82
`Syncing` (*cterasdk.edge.enum.SyncStatus attribute*), 82
`SyncStatus` (class in *cterasdk.edge.enum*), 80
`Syslog` (class in *cterasdk.edge.syslog*), 95
`System` (*cterasdk.core.enum.LogTopic attribute*), 59

T

`TakingSnapshot` (*cterasdk.edge.enum.SyncStatus attribute*), 82
`target_host()` (*cterasdk.edge.session.Session method*), 90
`Task` (class in *cterasdk.edge.taskmgr*), 96
`TaskError`, 96
`TaskStatus` (class in *cterasdk.edge.enum*), 82
`TaskStatusEnum` (class in *cterasdk.edge.taskmgr*), 96
`TCP` (*cterasdk.edge.enum.IPProtocol attribute*), 78
`tcp_connect()` (*cterasdk.edge.network.Network method*), 86
`TCPConnectRC` (class in *cterasdk.edge.enum*), 82
`Telnet` (class in *cterasdk.edge.telnet*), 96
`TempfileServices` (class in *cterasdk.lib.tempfile*), 101
`tenant()` (*cterasdk.core.session.Session method*), 64
`tenant()` (*cterasdk.edge.session.RemoteSession method*), 90
`tenants()` (*cterasdk.core.portals.Portals method*), 62
`terminate()` (in module *cterasdk.core.session*), 65
`terminate()` (in module *cterasdk.edge.session*), 90
`test()` (*cterasdk.object.Gateway.Gateway method*), 103
`test()` (*cterasdk.object.Portals.Portals method*), 105
`test()` (in module *cterasdk.core.connection*), 55
`test()` (in module *cterasdk.edge.connection*), 74

`test_application()` (in module `cterasdk.edge.connection`), 74
`test_conn()` (`cterasdk.client.host.NetworkHost` method), 47
`test_network()` (in module `cterasdk.core.connection`), 55
`test_network()` (in module `cterasdk.edge.connection`), 74
`textplain` (`cterasdk.client.http.ContentType` attribute), 47
`tid()` (`cterasdk.edge.taskmgr.Task` static method), 96
`Timezone` (class in `cterasdk.edge.timezone`), 97
`to_server_object()` (`cterasdk.edge.types.ShareAccessControlEntry` method), 97
`to_server_object()` (`cterasdk.edge.types.UserGroupEntry` method), 98
`tojsonstr()` (in module `cterasdk.convert.format`), 49
`toxml()` (in module `cterasdk.convert.format`), 49
`toxmlstr()` (in module `cterasdk.convert.format`), 49
`track()` (`cterasdk.lib.tracker.StatusTracker` method), 101
`track()` (in module `cterasdk.lib.tracker`), 101
`transcribe()` (in module `cterasdk.transcript.transcribe`), 105
`TraverseFolderExecuteFile` (`cterasdk.edge.enum.AuditEvents` attribute), 76
`trust()` (`cterasdk.client.http.HttpClientBase` method), 48
`type_attr` (`cterasdk.core.devices.Devices` attribute), 57

U

`UDP` (`cterasdk.edge.enum.IPProtocol` attribute), 78
`undeleate()` (`cterasdk.core.cloudfs.CloudFS` method), 55
`undeleate()` (`cterasdk.core.files.browser.FileBrowser` method), 51
`undeleate()` (`cterasdk.core.portals.Portals` method), 62
`undeleate()` (in module `cterasdk.core.files.recover`), 53
`undeleate_multi()` (`cterasdk.core.files.browser.FileBrowser` method), 51
`undeleate_multi()` (in module `cterasdk.core.files.recover`), 53
`union()` (in module `cterasdk.core.union`), 66
`Unknown` (`cterasdk.edge.enum.VolumeStatus` attribute), 83
`Unlicensed` (`cterasdk.edge.enum.BackupConfStatusID` attribute), 76
`Unlicensed` (`cterasdk.edge.enum.SyncStatus` attribute), 82
`UNLIKE` (`cterasdk.core.query.Restriction` attribute), 64
`Unmounted` (`cterasdk.edge.enum.VolumeStatus` attribute), 83
`unpin_all()` (`cterasdk.edge.cache.Cache` method), 73
`Unsubscribed` (`cterasdk.edge.enum.BackupConfStatusID` attribute), 77
`unsuspend()` (`cterasdk.edge.backup.Backup` method), 71
`unsuspend()` (`cterasdk.edge.sync.Sync` method), 95
`update_current_tenant()` (in module `cterasdk.core.decorator`), 55
`update_tenant()` (`cterasdk.core.session.Session` method), 64
`UpgradingDataBase` (`cterasdk.edge.enum.SyncStatus` attribute), 82
`upload` (`cterasdk.edge.support.DebugLevel` attribute), 94
`upload()` (`cterasdk.client.cteraclient.CTERAClient` method), 45
`upload()` (`cterasdk.client.host.CTERAHost` method), 46
`upload()` (`cterasdk.client.http.HTTPClient` method), 47
`upload()` (`cterasdk.core.files.browser.FileBrowser` method), 51
`upload()` (`cterasdk.edge.files.browser.FileBrowser` method), 68
`upload()` (`cterasdk.lib.file_access_base.FileAccessBase` method), 100
`urlencoded` (`cterasdk.client.http.ContentType` attribute), 47
`User` (`cterasdk.core.enum.PortalAccountType` attribute), 59
`user_name()` (`cterasdk.core.session.Session` method), 64
`UserAccount` (class in `cterasdk.core.types`), 65
`UserGroupEntry` (class in `cterasdk.edge.types`), 97
`username()` (`cterasdk.edge.session.ActiveSession` method), 90
`Users` (class in `cterasdk.core.users`), 66
`Users` (class in `cterasdk.edge.users`), 98
`Users` (`cterasdk.core.enum.SearchType` attribute), 60
`XML` (`cterasdk.convert.xml_types.XMLTypes` attribute), 50

V

`VAL` (`cterasdk.convert.xml_types.XMLTypes` attribute), 50
`validate_directory()` (`cterasdk.lib.filesystem.FileSystem` method), 100
`Version` (class in `cterasdk.lib.version`), 102
`version()` (`cterasdk.lib.filesystem.FileSystem` static method), 100

vGateway (*cterasdk.core.enum.DeviceType* attribute),
58
 Volumes (*class in cteraskd.edge.volumes*), 99
 VolumeStatus (*class in cteraskd.edge.enum*), 82
 VolumeUnavailable
 (*cterasdk.edge.enum.SyncStatus* attribute),
82

W

wait() (*cterasdk.edge.power.Boot* method), 87
 wait() (*cterasdk.edge.taskmgr.Task* method), 96
 wait() (*in module cteraskd.edge.taskmgr*), 96
 walk() (*cterasdk.core.files.browser.FileBrowser*
method), 51
 WARNING (*cterasdk.core.enum.Severity* attribute), 60
 WARNING (*cterasdk.edge.enum.Severity* attribute), 80
 warning (*cterasdk.edge.support.DebugLevel* attribute),
95
 whoami() (*cterasdk.client.host.CTERAHost* method),
46
 whoami() (*cterasdk.core.session.Session* method), 64
 WindowsNT (*cterasdk.edge.enum.Acl* attribute), 75
 WorkstationAgent (*cterasdk.core.enum.DeviceType*
attribute), 58
 write() (*cterasdk.lib.filesystem.FileSystem* static
method), 100
 WriteAttributes (*cterasdk.edge.enum.AuditEvents*
attribute), 76
 WriteExtendedAttributes
 (*cterasdk.edge.enum.AuditEvents* attribute), 76
 WrongPassword (*cterasdk.edge.enum.BackupConfStatusID*
attribute), 77

X

XMLTypes (*class in cteraskd.convert.xml_types*), 49

Z

Zones (*class in cteraskd.core.zones*), 67