Designsafe-CI API Overview

DesignSafe-CI provides full scriptable access to its underlying infrastructure via the Agave API, which provides a comprehensive set of RESTful web services that make it easy for developers and users to:

- Develop and run applications on HPC, Cloud, Condor, and container-based computing systems
- Use MyProxy-based authentication for federated identity
- Bring their own computing and storage resources into Designsafe
- Share data and applications, even with people who aren't Designsafe users
- Connect computing and data tasks via web-based events
- Manage data on any cloud storage platform one has access to
- Build sophisticated web-based applications that take advantage of all these underlying capabilities

Initial Assumptions

- You have access to the Designsafe-CI.org portal and you know your username/password for it
- You have access to a Linux or Mac OS X commmand-line environment. On Windows, you can work inside a virtual machine configured with any variant of Linux.
- You are comfortable editing text files and working at the command line
- You have Git, Python 2.7.6+, curl, and Bash installed

Installing the CLI

The Agave API comes bundled with a set of command line scripts for interacting with it. Using these scripts is generally easier than hand-crafting cURL commands, but if you prefer that route, consult Getting Started with the Agave API. All tutorials presented here will assume you are using the command line tools.

On the target system, cd to whatever directory you would like to install the CLI into. We suggest putting it in your HOME directory. Enter the following commands:

```
cd $HOME
```

or

```
cd ~
```

Now, clone the `foundation-cli` project using `git` and add the `bin` directory to your `PATH`

```
git clone https://bitbucket.org/taccaci/foundation-cli.git agave-cli
export PATH=$PATH:$HOME/agave-cli/bin
```

You may want to make that modification to your PATH permanent by adding this line to your `.bashrc` file

```
echo "export PATH=\$PATH:\$HOME/agave-cli/bin" >> ~/.bashrc
```

Now, initialize the CLI to work with DesignSafe-Ci

```
tenants-init -t designsafe
```

You should get this response:

```
You are now configured to interact with the APIs at https://agave.designsafe-ci.org/
```

This completes the section on installing the DesignSafe CLI

Creating a Oauth2 client

Designsafe's APIs use OAuth2 for managing authentication and authorization. Before you start working with these APIs, you will need to create a OAuth client application associated with a set of API keys. This is a one-time action, so if you already have a set of API keys, skip to the next step. If not, you can create your keys using the clients service as follows:

```
clients-create -S -v -N my_client -D "Client used for app development"
```

Note: The -N flag allows you to specify a machine-readable name for your application; -D provides the description, and -S option stores your API keys for future use, so you will not need to manually enter them when you authenticate later.

After being prompted for your Designsafe username and password, you should get a response from clients-create that looks similar to this:

```
{
    "_links": {
        "self": {
            "href": "https://agave.designsafe-ci.org/clients/v2/my_client"
        },
        "subscriber": {
            "href": "https://agave.designsafe-ci.org/profiles/v2/username"
        },
        "subscriptions": {
            "href": "https://agave.designsafe-
ci.org/clients/v2/my_client/subscriptions/"
        }
    },
    "callbackUrl": "",
    "consumerKey": "fMYfhijQE8ajqcKaswlGH4D5sngh",
    "consumerSecret": "8FZ6K9QwN1PY6bA9SOcyi4oaEkMa",
    "description": "Client used for app development",
    "name": "my_client",
    "tier": "Unlimited"
}
```

Although much of the process of interacting with the Agave API is automated, you may need access to the `consumerKey`and `consumerSecret` for other types of OAuth2-based interaction, so please record them. If you lose them, you can create new copy of them for the same client by deleting the old client and creating it again. You can also have multiple OAuth2 clients - we are simply demonstrating use of one.
This completes the section on obtaining a set of API keys.

Obtaining and refreshing OAuth2 tokens

Tokens are a form of short-lived, temporary authenticiation and authorization used in place of your username and password when interacting with Designsafe APIs. Designsafe tokens expire 1 hour, but can easily be refreshed.

Obtain a token

```
# From your terminal interface, type:
auth-tokens-create -S -v
```

You will be prompted to enter your *API password*. Type your Designsafe password. Next, you will receive an affirmation of success in your terminal.

```
Token for designsafe:username successfully refreshed and cached for 3600 seconds
{
    "access_token": "96797ebac3e648b9453dfe4c57b7c0",
    "expires_in": 3600,
    "refresh_token": "cc5b3dff98a930f9207e148bb22f8",
    "scope": "default",
    "token_type": "bearer"
}
```

Refresh your token

When your token expires in an hour, you may refresh it:

```
auth-tokens-refresh -S -v
Token for designsafe:username successfully refreshed and cached for 3600 seconds
{
    "access_token": "3baebe5418ffce0da7fbdcb193d0ef",
    "expires_in": 3600,
    "refresh_token": "4a51a7524638e9d7ed0c3adcd3e99d5",
    "scope": "default",
    "token_type": "bearer"
}
```

This topic is covered in great detail at Authentication Token Management in the Agave live docs

This completes the section on obtaining an OAuth2 authentication token.

Designsafe Files tutorial

Working with files and folders in Designsafe's Files service works the same regardless of the type, location, or protocols used by the underlying storage.

Browsing

Here's how to list the home directory of a specific `username`

```
files-list username
.
octocat.png
```

That was the summary view. View all attributes of every file or folder by adding the verbose flag: `files-list -v username`

```
[
    {
        "_links": {
            "history": {
                "href": "https://agave.designsafe-
ci.org/files/v2/history/system/designsafe.storage.default/username"
            },
            "metadata": {
                "href": "https://agave.designsafe-
ci.org/meta/v2/data?q={\"associationIds\":\"7044784514159415781-242ac112-0001-002\"}"
            },
            "self": {
                "href": "https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username"
            },
            "system": {
                "href": "https://agave.designsafe-
ci.org/systems/v2/designsafe.storage.default"
            }
        },
        "format": "folder",
        "lastModified": "2016-02-28T14:07:58.000-06:00",
        "length": 32768,
        "mimeType": "text/directory",
        "name": ".",
        "path": "username",
        "permissions": "ALL",
        "system": "designsafe.storage.default",
        "type": "dir"
    },
    {
        "_links": {
            "self": {
                "href": "https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/octocat.png"
```

```
            },
            "system": {
                "href": "https://agave.designsafe-
ci.org/systems/v2/designsafe.storage.default"
            }
        },
        "format": "raw",
        "lastModified": "2016-02-28T14:07:59.000-06:00",
        "length": 3987,
        "mimeType": "application/octet-stream",
        "name": "octocat.png",
        "path": "username/octocat.png",
        "permissions": "READ_WRITE",
        "system": "designsafe.storage.default",
        "type": "file"
    }
]
```

Find out details about a specific file or folder by specifying its path: `files-list username/octocat.png`

Uploading

You can upload data into Designsafe storage by performing a multipart POST on the FILES service. Using the CLI, recursive directory uploads are supported. If you are manually calling `curl`, you will need to manually create the directories and upload the local contents one at a time. You can take a look in the `files-upload` script to see how this is done. Let's go ahead and upload a file to use in the rest of the tutorial.

```
files-upload -v -F lorem.txt username
```

Response:

```
{
    "_links": {
        "history": {
            "href": "https://agave.designsafe-
ci.org/files/v2/history/system/designsafe.storage.default/username/lorem.txt"
        },
        "self": {
            "href": "https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/lorem.txt"
        },
        "system": {
            "href": "https://agave.designsafe-
ci.org/systems/v2/designsafe.storage.default"
        }
    },
    "internalusername": null,
    "lastModified": "2016-02-28T14:20:51.386-06:00",
    "name": "lorem.txt",
    "nativeFormat": "raw",
    "owner": "username",
    "path": "username/lorem.txt",
    "source": "http://172.17.0.1/lorem.txt",
    "status": "STAGING_QUEUED",
    "systemId": "designsafe.storage.default",
    "uuid": "4401627548025875995-242ac112-0001-002"
}
```

If you specify `-N name` when uploading, the file will be renamed as it is sent to the storage resource. For instance, `-N ipsum.txt`.

Importing data from a URL

You can also import data from an external URL, so long as it is available without authentication. Rather than making a multipart file upload request, you can pass in JSON object with the URL and an optional target file name, file type, and array of notifications which should be made when the import completes. The next example will import a the README.md file from a public git repository in Bitbucket.

```
files-import -v -U "https://bitbucket.org/taccaci/agave-samples/raw/master/README.md"
username
```

Response:

```
{
    "_links": {
        "history": {
            "href": "https://agave.designsafe-
ci.org/files/v2/history/system/designsafe.storage.default/username/README.md"
        },
        "self": {
            "href": "https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/README.md"
        },
        "system": {
            "href": "https://agave.designsafe-
ci.org/systems/v2/designsafe.storage.default"
        }
    },
    "internalusername": null,
    "lastModified": "2016-02-28T14:28:36.759-06:00",
    "name": "README.md",
    "nativeFormat": "raw",
    "owner": "username",
    "path": "username/README.md",
    "source": "https://bitbucket.org/taccaci/agave-samples/raw/master/README.md",
    "status": "STAGING_QUEUED",
    "systemId": "designsafe.storage.default",
    "uuid": "2869171603221442075-242ac112-0001-002"
}
```

Notes: Importing data from a third party is done as an asynchronous activity. This means that while you will get a response from the API server right away, it may take some time before the data is available in the system. This means if you perform a file-list on the file as it is being imported, the value you get back may be incorrect. In this exercise, the file was just a few KB, so you should see it appear in your home folder almost immediately. If you were importing larger datasets, the transfer could take significantly longer depending on network quality and resource availability. In this case, you would see the file size continue to increase until it completed. In the event of a failed transfer, the files API will retry 3 times before canceling the transfer.

File operations

Similar to a POSIX filesysten, we can create, copy, move, rename, and delete files and folders. Let's try these out on one of the files we just uploaded.

```
$ files-mkdir -N newfolder username
Successfully created folder newfolder

$ files-list username
.
lorem.txt
newfolder
README.md

$ files-copy -D username/newfolder/lorem.txt username/lorem.txt
Successfully copied username/lorem.txt to username/newfolder/lorem.txt

$ files-list username/newfolder
.
lorem.txt

files-move -D username/newfolder/lorem2.txt  username/newfolder/lorem.txt
Successfully moved username/newfolder/lorem.txt to username/newfolder/lorem2.txt

$ files-list username/newfolder
.
lorem2.txt

$ files-rename -N lorem.txt username/newfolder/lorem2.txt
Successfully renamed username/newfolder/picsumipsum2.txt to
username/newfolder/picsumipsum.txt

$ files-delete username/lorem.txt
# Warning: The files API will perform recursive deletion on folders, so be careful!
Successfully deleted username/lorem.txt
```

```
$ files-list username
.
newfolder
README.md
```

Provenance

Designsafe keeps track of the provenance and history of every file object under its purview. Here's an example showing the provenance of the `lorem2.txt` file we just created.

```
files-history -v username/newfolder/lorem2.txt
```

Response:

```
[
    {
        "created": "2016-02-28T14:36:25.000-06:00",
        "description": "File item copied from https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/newfolder/lorem.txt"
,
        "status": "CREATED"
    },
    {
        "created": "2016-02-28T14:38:02.000-06:00",
        "description": "Moved from https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/newfolder/lorem.txt
to https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/newfolder/lorem2.txt
",
        "status": "MOVED"
    }
]
```

Downloading

You can easily download a file or folder from Designsafe's storage. As with uploads, you can specify a flag to rename the file on the fly.

```
files-get -N downloaded.txt username/newfolder/lorem2.txt
######################################################################## 100.0%
```

Add the recursive flag `--recursive` to download a directory.

Multiple Systems

Designsafe is architected around providing federated access to multiple file storage locations. By default, user files are stored in `designsafe.storage.default` and the public NEES datasets are found in `nees.public`. Others may be added to the system as the project progresses. To list all available storage resources at Designsafe, use the `systems-list` command.

```
systems-list -S
nees.public
designsafe.storage.boxsync
designsafe.storage.default
```

If you look at the tutorial material above, you'll notice that all file operations we've done to date were on the`designsafe.storage.default` system.

Working with other storage systems

Let's use a specific NEES data set on the `nees.public` system as an example.
List all folders (this is the equivalent to visiting `https://designsafeci-dev.tacc.utexas.edu/data/public/` in your browser).

```
files-list -S nees.public /

.

examples
facility.groups
go
nees
NEES-2005-0002.groups
NEES-2005-0004.groups
NEES-2005-0006.groups
...
```

List a specific directory:

```
files-list -S nees.public /facility.groups/UTexas
.
Contact
EducationOutreach
Equipment
neesUTexas1p.pdf
sensors
TrainingAndCertification
```

Download a specific file to your local system:

```
files-get -S nees.public /facility.groups/UTexas/neesUTexas1p.pdf
```

```
######################################################################## 100.0%
```

Import a specific file from `nees.public` into your Designsafe workspace

```
files-import -v -U "agave://nees.public/facility.groups/UTexas/neesUTexas1p.pdf" -S
designsafe.storage.default username
```

Response:

```
{
    "_links": {
        "history": {
            "href": "https://agave.designsafe-
ci.org/files/v2/history/system/designsafe.storage.default/username/neesUTexas1p.pdf"
        },
        "self": {
            "href": "https://agave.designsafe-
ci.org/files/v2/media/system/designsafe.storage.default/username/neesUTexas1p.pdf"
        },
        "system": {
            "href": "https://agave.designsafe-
ci.org/systems/v2/designsafe.storage.default"
        }
    },
    "internalUsername": null,
    "lastModified": "2016-02-28T15:04:04.027-06:00",
    "name": "neesUTexas1p.pdf",
    "nativeFormat": "raw",
    "owner": "username",
    "path": "username/neesUTexas1p.pdf",
    "source": "agave://nees.public/facility.groups/UTexas/neesUTexas1p.pdf",
    "status": "STAGING_QUEUED",
    "systemId": "designsafe.storage.default",
    "uuid": "3734420596388392475-242ac112-0001-002"
}
```

Result:

```
files-list -S designsafe.storage.default username
.
neesUTexas1p.pdf
newfolder
README.md
```

Sharing files with Designsafe collaborators

Every digital object at Designsafe has fine-grained access controls, including all files and folders. Let's give read access to the lorem2.txt file with another Designsafe user `dan`

```
files-pems-update -U dan -P READ username/newfolder/lorem2.txt
```

```
Successfully updated permission for dan
```

Verify that permissions were set:

```
files-pems-list username/newfolder/lorem2.txt
dan READ
username READ WRITE EXECUTE
```

Now, revoke that access grant...

```
$ files-pems-update files-pems-update -U dan -P NONE username/newfolder/lorem2.txt
Successfully updated permission for dan

$ files-pems-list username/newfolder/lorem2.txt
username READ WRITE EXECUTE
```