Working with Persistent Identifiers - B2HANDLE Hands-on - Demo

This lecture takes you through the steps to create and administer PIDs employing the B2HANDLE python library. We will create two PIDs for the local and public example file from the CURL path and link them by their PIDs.

We will cover:

- Authentication with the Handle server with certificates
- Registering a file
- Modifying and updating Handles
- Reverse look-ups

For resolving PIDs please use: `http://hdl.handle.net/`

Preparing the credentials file

B2HANDLE uses a json file to authenticate with the Handle server. You will find an example file in *HandleCerts*:

```
cat cred_21.T12995.json

{
    "handle_server_url": "https://epic4.storage.surfsara.nl:8007",
    "baseuri": "https://epic4.storage.surfsara.nl:8007",
    "private_key": "/home/ubuntu/HandleCerts/308_21.T12995_USER01_privkey.pem",
    "certificate_only":
    "/home/ubuntu/HandleCerts/308_21.T12995_USER01_certificate_only.pem",
    "prefix": "21.T12995",
    "handleowner": "200:0.NA/21.T12995",
    "reverselookup_username": "21.T12995",
    "reverselookup_password": "5f22530b03a4f024c0422e24b6a27eac",
    "HTTPS_verify": "False"
}
```

It contains the same information that we used for the CURL part: Handle server, private key, certificate, prefix and the reverse lookup password.

The handle owner is predefined, since the B2HANDLE library automatically creates the key-value pair at index 100 defining who is the owner with which rights.

The certficate verufucation is suppressed by setting *HTTPS_verify* to *False*.

Hands-on

- Create a copy of this file in your *home* directory
- Adjust the path to the certificates

Python basics

- Start ipython

  ```
  ipython
  ```

  ipython is a python interpreter with syntax highlighting and tab completion. You can test code line by line. It is very suitable for exploring new python packages.

- Load libraries Let me first demonstrate how to load The command for loading standard libraries and modules is *import*.

  ```
  # System libraries
  import uuid
  import hashlib
  import os, shutil
  ```

  If you want to use single function from a module or library use the *from … import* construct.

  ```
  # B2HANDLE
  from b2handle.clientcredentials import PIDClientCredentials
  from b2handle.handleclient import EUDATHandleClient
  ```

- ipython offers tab-completion:

  - Type in *from b2handle.* and press tab.
  - Type in *from b2handle.clientcredentials import* and press tab.

Exercise

Load the libraries and objects using tab completion. Now we have the objects *uuid*, *hashlib*, *os*, *shutil* and the b2handle objects *PIDClientCredentials* and *EUDATHandleClient* at our disposal. Check with *who* which variables are loaded into our namespace (ipython specific).

```
who
EUDATHandleClient    PIDClientCredentialshashlib     os   shutil   uuid
```

Connect to the Handle server

```
cred = PIDClientCredentials.load_from_JSON('<full_path>/<to_crdentials>.json')
ec = EUDATHandleClient.instantiate_with_credentials(cred)
```

You can also use tab-completion to complete the path to the credentials file.

Register the global file

- Define the URL

  ```
  fileLocation = 'https://ndownloader.figshare.com/files/2292172'
  ```

- Create the Handle

  ```
  uid = uuid.uuid1()
  print(uid)
  print(type(uid))
  pid = cred.get_prefix() + '/' + str(uid)
  print(pid)
  ```

- Create the Handle entry

  ```
  Handle = ec.register_handle(pid, fileLocation)
  print(Handle)
  ```

- Check the Handle record

  Go to `http://hdl.handle.net/<your PID>?noredirect` . B2HANDLE automatically creates the *HS_ADMIN* entry with the parameters given in the json file.

- Register the file

  ```
  Handle = ec.register_handle(pid, fileLocation)
  ```

  B2HANDLE checks whether this Handle is already assigned.

Register the local file

With `history` you can review all of your previous commands. Create a Handle for the local file. **Watch out**, do not overwrite the preivous variable *Handle*!

```
fileLocation = '/home/ubuntu/surveys.csv'
uid = uuid.uuid1()
print(uid)
print(type(uid))
pid = cred.get_prefix() + '/' + str(uid)
print(pid)

Handle_loc = ec.register_handle(pid, fileLocation)
```

```
print(Handle)
print(Handle_loc)
```

Link the two Handles

With the function `ec.modify_handle_value` we can add and overwrite Handle entries.

Add the checksum and modify the Handle again:

```
md5val = hashlib.md5('surveys.csv').hexdigest()
ec.modify_handle_value(Handle, add_if_not_exist=True,
    **dict([('REPLICA', Handle_loc), ('MD5', md5val)]))
```

Now insert the link in the Handle entry pointing to the local file:

```
ec.modify_handle_value(Handle_loc, add_if_not_exist=True,
    **dict([('ORIGINAL', Handle), ('MD5', md5val)]))
```

Get Handle record

```
ec.retrieve_handle_record(<PID>)
```

Reverse look-ups

```
args = dict([('MD5', md5val)])
result = ec.search_handle(**args)
```

or

```
ec.search_handle('www.test.com')
```

Reverse look-ups are restricted in B2HANDLE. You can only search on the keys ‘URL’ or ‘CHECKSUM’.