# Working with Persistent Identifiers - CURL Hands-on

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

We will cover:

- Authentication with the Handle server
- Registering a file using PUT
- Retrieving data and metdata with GET
- Modifying and updating Handles
- Reverse look-ups

## Main Parameters of CURL

The main command `curl [options] [URL...]`

Before we start, we explain the main parameters of CURL used as options

- **-X, –request** : (HTTP) Specifies a custom request method to use when communicating with the HTTP server. The specified request method will be used instead of the method otherwise used (which defaults to GET). Common additional HTTP requests include PUT ,POST and DELETE . ( -X GET)
- **-U, –proxy-user <user:password>** : Specify the user name and password to use for proxy authentication. (ex: -u :)
- **-H, –header**
  : Extra header to include in the request when sending HTTP to a server. You may specify any number of extra headers. (ex: -H “Accept: application/json” so as to accept json data)
- **-d, –data** : (HTTP) Sends the specified data in a POST or PUT request to the HTTP server, in the same way that a browser does when a user has filled in an HTML form and presses the submit button.
- **-D, –dump-header** : Write the protocol headers to the specified file.
- **-v**: get verbosed information about the connection with the server

These are the main parameters we are going to use in our examples. For more parameters please check cURL.

## Connect to the SURFsara handle server

To connect to the handle server you need to provide a **prefix** and its respective **private key** and **certificate**, the latter two are pem-files. We will use the prefix *21.T12995*, a test prefix at SURFsara. Since we use

these parameters everytime we call curl, it is handy to store them in shell variables:

```
PRIVKEY=HandleCerts/308_21.T12995_TRAINING_privkey.pem
CERTIFICATE=HandleCerts/308_21.T12995_TRAINING_certificate_only.pem
PID_SERVER=https://epic4.storage.surfsara.nl:8007/api/handles
PREFIX=21.T12995
```

# Registering a file with PUT

## Building the PID:

- Create a universally unique identifier (uuid)
- Take the function *uuidgen* for this

```
SUFFIX=`uuidgen`
```

- Concatenate your PID prefix and the uuid to create the full PID `$PREFIX/$SUFFIX`

Since the prefix is unique and we employed a uuid to create the suffix, we now have an opaque string which is unique to our resolver ($PREFIX/$SUFFIX).

The URL in the CURL request is:

```
https://epic4.storage.surfsara.nl:8007/api/handles/$PREFIX/$SUFFIX
```

or when you set the variable *PID_SERVER*

```
$PID_SERVER/$PREFIX/$SUFFIX
```

## Creating the handle entry

Register the PID, link the PID and the data object. Here we use a csv file as data which is stored on figshare and publicly available.

```
SUFFIX=`uuidgen`

curl -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data \
        '{"values": [
            {"index":1,"type":"URL",
                "data": {"format": "string",
                "value":"https://ndownloader.figshare.com/files/2292172"}},
            { "index": 100,"type": "HS_ADMIN",
                "data": {"format": "admin",
                "value": {"handle": "0.NA/'$PREFIX'","index":
                200,"permissions": "011111110011"}}}
        ]}' \
$PID_SERVER/$PREFIX/$SUFFIX | python -m json.tool
```

This gives the response:

```
{"responseCode":1,"handle":"PREFIX/AE919576-226E-412D-BC9D-73682DD207F5"}
```

indicating, that the handle was created.

**Note**, that we suppressed the server certificate verification here with the option '-k'. For security reasons you should not use this option with real prefixes.

# Resolving handles, retrieving data and metadata

Let us go to the resolver and see what is stored there:

```
http://hdl.handle.net
```

We can get some information on the data from the resolver. We can retrieve the data object itself via the web-browser.

## Rerieve the handle record

You can retrieve the PID record using the *GET* option of curl from the local handle server directly

```
curl -k -X GET $PID_SERVER/$PREFIX/$SUFFIX | python -m json.tool
```

or from the global handle resolver

```
curl -k -X GET http://hdl.handle.net/api/handles/$PREFIX/$SUFFIX | python -m json.tool
```

Here we do not need to authorise since the handle record is public.

### Exercises: File and metadata retrieval (10 minutes)

- How can you retrieve the document behind the PID?
- Download the file via the resolver. Try to use *wget* when working remotely on our training machine.
- How is the data stored when downloading via the browser and how via *wget*?
- How can you retrieve the handle record in a web browser?
- Inspect the HS_ADMIN field**
- What happens if you try to reregister the file with the same PID?

### Solution

- `wget hdl.handle.net/21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff`
- http://hdl.handle.net/21.T12995/5c6172f0-d7e2-11e6-9fe4-fa163edb14ff?noredirect

## No overwriting of handles

We saw in the last exercise, that the data in the handles can be overwritten. That is useful in some cases, as we will see later. However, upon registration you might want to check that you do not overwrite existing data. A secure way to create handles is:

```
curl -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data \
        '{"values":[
            {"index":1,"type":"URL",
                "data": {"format": "string",
                "value":"https://ndownloader.figshare.com/files/2292172"}},
            {"index":100,"type":"HS_ADMIN",
                "data":{"value":{"index":200,"handle":"0.NA/'$PREFIX'",
                "permissions":"011111110011","format":"admin"},
                "format":"admin"}}
        ]}' \
$PID_SERVER/$PREFIX/$SUFFIX?overwrite=false | python -m json.tool
```

This will return the response *101*, inidicating that the handle already exists.

# Modify handles and store additional data

## Exercise: Data and metdadata (5 minutes)

Reregister the file with some more metadata:

- At index 2 insert the key “ORIGIN” with value “Data Carpentry pandas example file”.
- At index 101 insert the key “FORMAT” with value “CSV”

## Solution

```
curl -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data \
        '{"values":[
            {"index":1,"type":"URL",
                "data": {"format": "string",
                "value":"https://ndownloader.figshare.com/files/2292172"}},
            {"index":2,"type":"DATATYPE",
                "data": {"format": "string",
                "value":"Data Carpentry pandas example file"}},
            {"index":100,"type":"HS_ADMIN",
                "data":{"value":{"index":200,"handle":"0.NA/'$PREFIX'",
                "permissions":"011111110011","format":"admin"},
                "format":"admin"}},
            {"index":101,"type":"FORMAT",
                "data":"csv"}]}' \
$PID_SERVER/$PREFIX/$SUFFIX | python -m json.tool
```

# Update the Handle record

```
MD5VALUE=` md5 surveys.csv | awk '{ print $4 }'`

curl -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data '{"index":3, "type":"MD5",
    "data": {"format": "string","value": "'$MD5VALUE'"}}' \
$PID_SERVER/$PREFIX/$SUFFIX?index=3 | python -m json.tool
```

# Exercise: Update the Handle record (10min)

1. Create the checksum of the file and store it at index 3 WITHOUT reregistering the file (tip use '?index=')
2. Update the Handle record in one go with: Overwriting index 2 Creating index 4 with key = "SIZE" and the appropriate value Creating index 5 with key = "FORMAT" and value "CSV"
3. Use curl's DELETE option to delete index 101. Watch out: do not delete the whole Handle

### Solution ad 2.

```
curl -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data '{"values": [
        {"index": 2, "type": "DATATYPE",
        "data": {"format": "string","value": "PID training example"}},
        {"index": 4, "type": "SIZE",   "data": {"format": "string","value": "N/A"}},
        {"index": 5, "type": "FORMAT", "data": {"format": "string","value": "csv"}}
        ]}' \
    $PID_SERVER/$PREFIX/$SUFFIX?index=2\&index=4\&index=5 | python -m json.tool
```

### Solution ad 3.

```
$CURL -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X DELETE \
    $PID_SERVER/$PREFIX/$SUFFIX?index=101 | python -m json.tool
```

## Reverse look-ups

Given a certain entry in the PID record, how can you find the repsective PID? This feature is **NOT** part of the standard Handle API. Handle servers in the EUDAT domain offer a special reverse look-up servelet to facilitate this function. To authorise with the servelet you need a password with your prefix. The command has the following structure:

```
$PID_REV=https://epic4.storage.surfsara.nl:8007/hrls/handles
curl -k -u "21.T12995:<password>" $PID_REV?<KEYWORD>=*
```

The password can be found in *HandleCerts/cred_21.T12995.json*. **Note**, that you do not have to access the *api* on the handle server but the *handle reverse look-up servelet (hrls)*.

To get a list of PIDs under the handle instance:

```
curl -k -u "21.T12995:<password>" $PID_REV?URL=*
```

This shows the first 1000 PIDs on the server.

And you can set extra parameters:

```
$CURL -k -u "21.T12995:<password>" $PID_REV?URL=*\&limit=10
```

# Exercise: Linking files by PIDs (15min)

Let us label the downloaded copy of the csv file with a new PID. The file should reside in your download folder or at a location that you specified when using *wget* to download the file. Replace with the appropriate path and filename.

1. Register a local copy of the public file “https://ndownloader.figshare.com/files/2292172”
2. Store some additional information to check the integrity of the local data
3. Update the Handle records and mark one of the files as original file and the other as replica
4. Try to resolve to the local file. Does it work?

## Solution: Register local file

```
SUFFIX2=`uuidgen`
FILELOC=<PATH>
MD5SUM=` md5 $FILELOC | awk '{ print $4 }'`

$CURL -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data \
        '{"values":[
        {"index":1,"type":"URL",  "data": {"format": "string","value":"'$FILELOC'"}},
        {"index":2,"type":"TYPE", "data":
            {"format": "string","value":"Data Carpentry pandas example file"}},
        {"index":3,"type":"MD5",
            "data": {"format": "string","value":"'$MD5SUM'"}},
        {"index":100,"type":"HS_ADMIN",
                "data":{"value":{"index":200,"handle":"0.NA/'$PREFIX'",
                "permissions":"011111110011","format":"admin"},
                "format":"admin"}}
        ]}' \
$PID_SERVER/$PREFIX/$SUFFIX2 | python -m json.tool
```

We have now two PIDs one pointing to the public file

```
echo $PREFIX/$SUFFIX
```

and one pointing to our local copy of that public file

```
echo $PREFIX/$SUFFIX2
```

### Solution: Link local and public file

We will link the two files using the keywork *REPLICA* and we will use index 4.

```
$CURL -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data '{"index":4, "type":"REPLICA",
    "data": {"format": "string","value":"'$PREFIX'/'$SUFFIX2'"}}' \
    $PID_SERVER/$PREFIX/$SUFFIX?index=4 | python -m json.tool

$CURL -k --key $PRIVKEY --cert $CERTIFICATE \
    -H "Content-Type:application/json" \
    -H 'Authorization: Handle clientCert="true"' \
    -X PUT --data '{"index":4, "type":"REPLICA",
    "data": {"format": "string","value":"'$PREFIX'/'$SUFFIX'"}}' \
    $PID_SERVER/$PREFIX/$SUFFIX2?index=4 | python -m json.tool
```

# Exercise: The Handle resolution

How does the handle resolver actually know to which data object to resolve to? In the cases above we explicitely defined the field *URL* at index 1. What happens if index 1 contains different information, i.e. a different key. And what happens when the URL appears somewhere else?

Create a PID for www.google.com as done above, verify that the resolving works. In which of the following cases does the resolving still work and why?

1. Change the key “URL” at index 1 to *bogus*.
2. Change the value at index 1 to something that will not resolve.
3. Insert the key “URL” at some index with the valid URL.
4. Add the value www.yahoo.com at index 1. Where does the PID resolve to
5. Correct the key at index 1 to “URL”. Where does the PID resolve to?