PIDs in EUDAT

About

Documentation of the EUDAT options for Persistent Identification (PID) for site admins and community data managers.

Modified: 06 March 2017

Synopsis

In order to access a data object stored in EUDAT, an associated persistent identifier (PID) is needed. EUDAT has adopted Handle-based persistent identifiers. This document discusses the basics of acquiring a Handle identifier. It also discusses the Handle system REST API in use in EUDAT. Formerly the EPIC API was standard delivered to use. EUDAT is moving to the Handle system REST API.

N.B.: This page is maintained for the benefit of legacy EUDAT users; we recommend against new uses of the EPIC API. If you are deploying a new instance, please see the B2HANDLE page.

Introduction

The Handle System is a software infrastructure offering general purpose identifier resolution services. Using the open set of protocols that the Handle System provides, a distributed computer system can create and manage handles (identifiers) of digital resources. The same set of protocols allows a client to locate, query the metadata and access the data of the digital resource identified by a Handle.

An identifier in the Handle System is composed of a prefix, a slash character ('/') and a suffix. Just like with the Domain Name System, there is only one Handle System in the universe. Each Handle consists of a *prefix* and a *suffix*, separated by a slash "/" character, for example: `10916/Hello_World`. The prefix denotes a resolution subsystem (a handle server) while the identifier points to the Handle record local to that Handle Server. The handles are globally resolvable, e.g. using http://hdl.handle.net/.

EUDAT requires integration of Handle in your infrastructure. Before a community or data centre can create their own PIDs, they need to have a prefix. With the prefix acquired, there are two options: you can run your own Handle system; or you can pass the details of your prefix to EUDAT partner SURFsara to manage it on your behalf. An additional benefit of using the EUDAT systems is access to a REST API to manage your PID handles. The REST API which is run by SURFsara is called EPIC. The handle version 8 now also supports a JSON REST API. The consortium which provides this functionality is the pid consortium. It is important to note that the EPIC is totally compatible with the DOI system, and a PID (including the prefix) can also be interpreted from any DOI server, such as: http://dx.doi.org/. This guarantees that adoption of EPIC does not tie you in to a domain-specific, esoteric technology useful only on European infrastructures but instead gives you access to a global identifier, interchangeable with others of this type.

If you have decided that your community runs its own Handle system, you have two options: you can follow our documentation about deploying an EPIC service; or you can follow the user documentation and guidelines of Handle (although the EUDAT guidelines how to buy a prefix may still be useful to you). The combined installation of your own handle system and EPIC service is described on a github page, which forms part of EPIC API v2.

If you have decided to let SURFsara manage your PID, this document describes how to acquire a prefix and how to use the SURFsara service to manage EUDAT PIDs. It also includes a discussion of the advanced REST API to

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.

manipulate PIDs if needed.

How to request a Production Prefix

If you would like to obtain a prefix, please contact EUDAT.

Using the PID Web Service with the Handle system RESTFUL API

EUDAT is moving to the use of the handle system RESTFUL JSON API. It is different from the EPIC API. It only has been around since handle version 8. This has been implemented in 2016.

Documentation about the API is on the handle system website: http://www.handle.net/hnr_documentation.html. It is described in: http://www.handle.net/tech_manual/HN_Tech_Manual_8.pdf chapter 14.

There is also a python library which enables easy access to the handle system RESTFUL API: https://github.com/EUDAT-B2SAFE/B2HANDLE. Documentation is on: http://eudat-b2safe.github.io/B2HANDLE/

There is also a course on the usage of the RESTFUL API via curl:
https://github.com/EUDAT-Training/B2SAFE-B2STAGE-Training/blob/master/07...

Using the PID Web Service with the EPIC API

While the use of PID in EUDAT is mostly done by iRODS and other EUDAT services, it is some times needed to manually create, access and update PID records. SURFsara provides a REST API for this, a service available to all EUDAT sites that have associated their Handle prefix with EPIC.

The PID web service can be found at https://epic3.storage.surfsara.nl/v2/handles/
or https://epic4.storage.surfsara.nl/v2_A/handles . You'll be asked for a user-name and password. The password is the same as it was for each test user. The username has changed for this version of the EPIC service. In order to make use of this service, your site needs a Handle prefix and this needs to be associated with EPIC, following the documented procedure. The current list of prefixes for the EPIC service is as follows:

handle prefix	assigned to:	epic service assigned to prefix
10916	SURFsara	https://epic3.storage.surfsara.nl/v2/handles/
11095	MPI-PL	https://epic3.storage.surfsara.nl/v2/handles/
11096	RZG	https://epic3.storage.surfsara.nl/v2/handles/
11097	JUELICH	https://epic3.storage.surfsara.nl/v2/handles/
11099	INGV	https://epic4.storage.surfsara.nl/v2_A/handles/
11100	CINECA	https://epic3.storage.surfsara.nl/v2/handles/
11101	BSC	https://epic4.storage.surfsara.nl/v2_A/handles/
11112	SURFsara	https://epic3.storage.surfsara.nl/v2/handles/
11113	CSC	https://epic3.storage.surfsara.nl/v2/handles/
11114	LifeWatch	https://epic4.storage.surfsara.nl/v2_A/handles/
11137	CINES	https://epic3.storage.surfsara.nl/v2/handles/
11139	STFC	https://epic3.storage.surfsara.nl/v2/handles/
11140	SNIC (KTH)	
11161	EPCC	https://epic3.storage.surfsara.nl/v2/handles/

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.

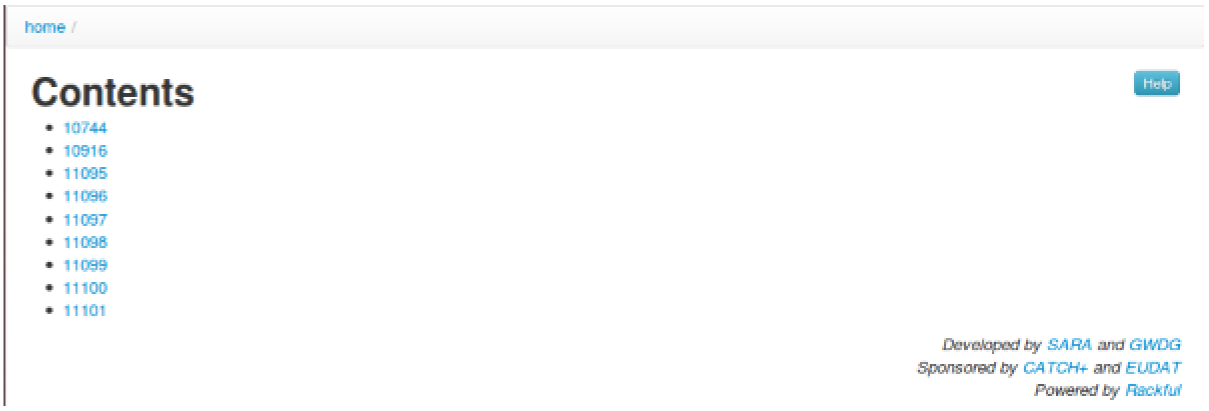
11202	SIGMA (University of Oslo)	https://epic3.storage.surfsara.nl/v2/handles/
11225	PSNC	https://epic4.storage.surfsara.nl/v2_A/handles
11230	KNMI	https://epic4.storage.surfsara.nl/v2_A/handles
11277	EBI/DIXA	https://epic4.storage.surfsara.nl/v2_A/handles
11304	CSC B2STORE	https://epic4.storage.surfsara.nl/v2_A/handles
11483	ELIXER	https://epic4.storage.surfsara.nl/v2_B/handles
11504	UVA	https://epic4.storage.surfsara.nl/v2_B/handles
11528	EBI	https://epic4.storage.surfsara.nl/v2_B/handles

The namespace of the Web Service is very simple:

```
https://epic3.storage.surfsara.nl/v2/handles/
┗ <prefix>/
 ┗ <suffix>
```

/v2/handles/

When you go to https://epic3.storage.surfsara.nl/v2/handles/ with a web browser, you'll see something like this



Contents

- 10744
- 10916
- 11095
- 11096
- 11097
- 11098
- 11099
- 11100
- 11101

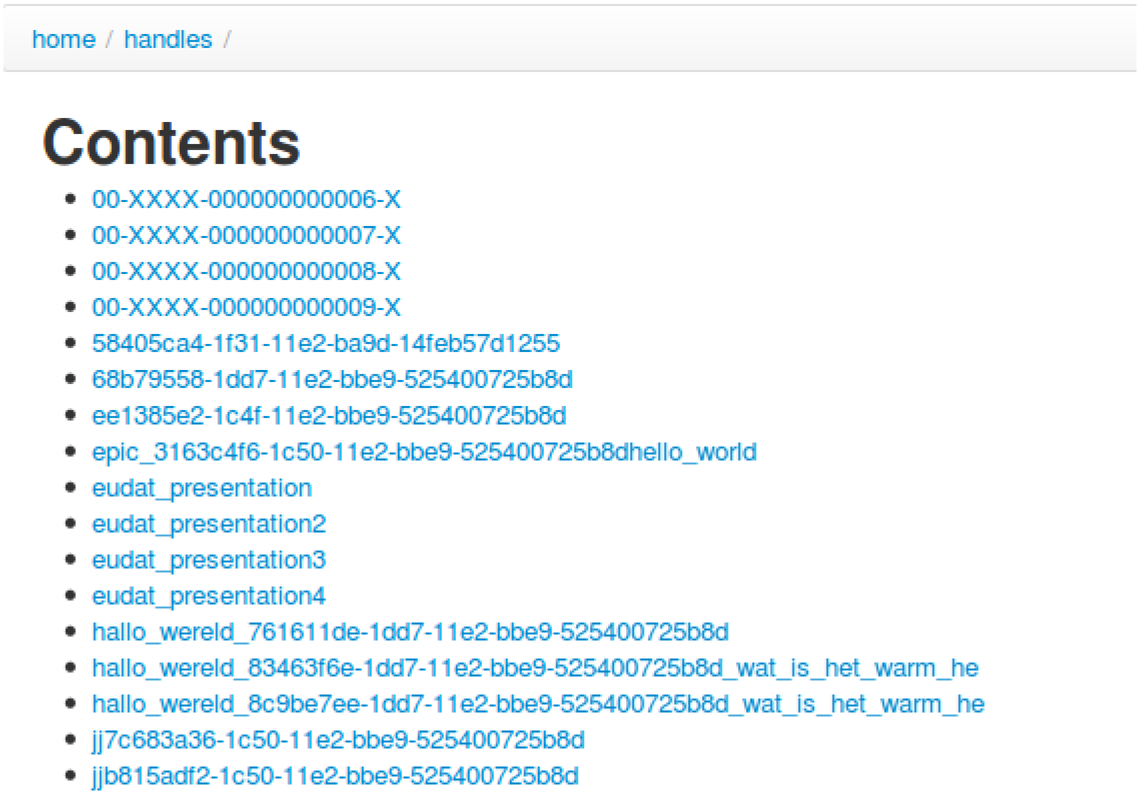
Developed by SARA and GWDG
Sponsored by CATCH+ and EUDAT
Powered by Rackful

This doesn't provide you with much information, but it demonstrates a few things: first of all, as you'd expect from a RESTful Web Service, you can "navigate" through the service namespace using hyperlinks. You're currently looking at (X)HTML, but the information is also available as JSON or as plain text.

/v2/handles/<prefix>

When you click on 10916/, you'll see something like this:

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.



home / handles /

Contents

- 00-XXXX-000000000006-X
- 00-XXXX-000000000007-X
- 00-XXXX-000000000008-X
- 00-XXXX-000000000009-X
- 58405ca4-1f31-11e2-ba9d-14feb57d1255
- 68b79558-1dd7-11e2-bbe9-525400725b8d
- ee1385e2-1c4f-11e2-bbe9-525400725b8d
- epic_3163c4f6-1c50-11e2-bbe9-525400725b8dhello_world
- eudat_presentation
- eudat_presentation2
- eudat_presentation3
- eudat_presentation4
- hallo_wereld_761611de-1dd7-11e2-bbe9-525400725b8d
- hallo_wereld_83463f6e-1dd7-11e2-bbe9-525400725b8d_wat_is_het_warm_he
- hallo_wereld_8c9be7ee-1dd7-11e2-bbe9-525400725b8d_wat_is_het_warm_he
- jj7c683a36-1c50-11e2-bbe9-525400725b8d
- jjb815adf2-1c50-11e2-bbe9-525400725b8d

On this page, you'll see a list of all Handles in this prefix. But you can also use this URL to search for certain Handles by their content. For example, to search for all Handles with a URL-field pointing at http://www.sara.nl/, use https://epic3.storage.surfsara.nl/v2/handles/10916/?URL=http://www.sara.nl/.

You can also use wildcards, with the asterisk "*" as wildcard character and the tilde "~" as escape character. Ie. if you want to search for a literal asterisk, you must specify "~*", and to search for a literal tilde, you must specify "~~". For example, if you want to search for all Handles pointing to an employee's personal page at SARA, use https://epic3.storage.surfsara.nl/v2/handles/10916/?URL=http://www.sara.nl/ ~~*&mode=wildcard.

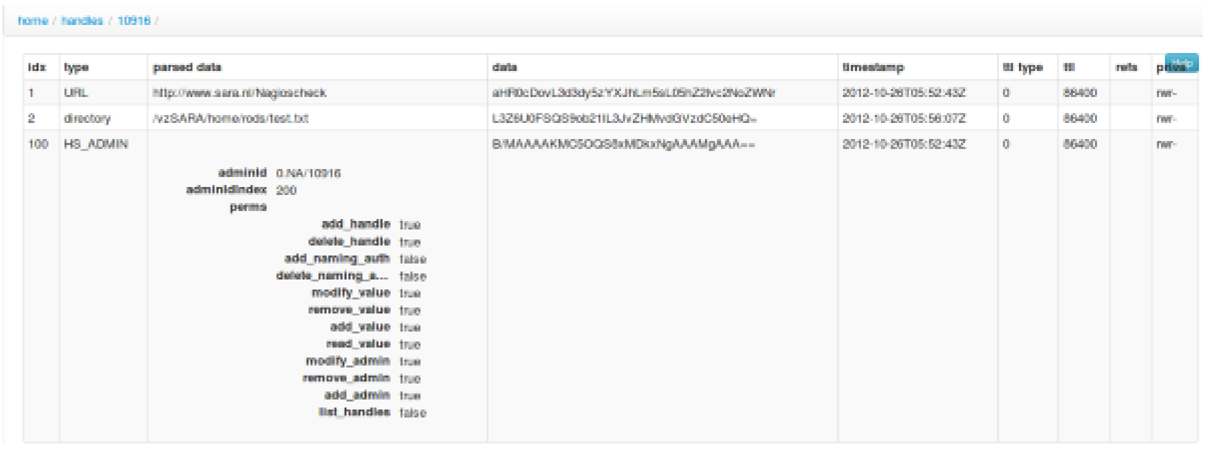
Again, this URL provides multiple representations.

/v2/handles/<prefix>/<suffix>

HTTP GET

When you click on 58405ca4-1f31-11e2-ba9d-14feb57d1255, you'll see something like this:

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.



home / handles / 10916 /

idx	type	parsed data	data	timestamp	ttl type	ttl	refs	privs
1	URL	http://www.sara.nl/Nagioscheck	aHR0cDovL3d3dy5zYXJhLm5sL05hZ2lvc2NoZWNr	2012-10-26T05:52:43Z	0	86400		rwr-
2	directory	/vzSARA/home/rods/test.txt	L3Z6U0FSQS9ob21lL3JvZHMvdGVzdC50eHQ=	2012-10-26T05:56:07Z	0	86400		rwr-
100	HS_ADMIN	adminId 0.NA/10916 adminIdIndex 200 perms add_handle true delete_handle true add_naming_auth false delete_naming_a... false modify_value true remove_value true add_value true read_value true modify_admin true remove_admin true add_admin true list_handles false	B/MAAAAKMC5OQS8xMDkxNgAAAMgAAA==	2012-10-26T05:52:43Z	0	86400		rwr-

Or, using JSON (prettified for readability):

```
[
  {
  "idx":1,
  "type":"URL",
  "parsed_data":"http://www.sara.nl/Nagioscheck",
  "data":"aHR0cDovL3d3dy5zYXJhLm5sL05hZ2lvc2NoZWNr",
  "timestamp":"2012-10-26T05:52:43Z",
  "ttl_type":0,
  "ttl":86400,
  "refs":[],
  "privs":"rwr-"
  },
  {
  "idx":2,
  "type":"directory",
  "parsed_data":"/vzSARA/home/rods/test.txt",
  "data":"L3Z6U0FSQS9ob21lL3JvZHMvdGVzdC50eHQ=",
  "timestamp":"2012-10-26T05:56:07Z",
  "ttl_type":0,
  "ttl":86400,
  "refs":[],
  "privs":"rwr-"
  },
  {
  "idx":100,
  "type":"HS_ADMIN",
  "parsed_data":{
    "adminId":"0.NA/10916",
    "adminIdIndex":200,
    "perms":{
       "add_handle":true,
       "delete_handle":true,
       "add_naming_auth":false,
       "delete_naming_auth":false,
```

EUDAT receives funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 654065.

```
       "modify_value":true,
       "remove_value":true,
       "add_value":true,
       "read_value":true,
       "modify_admin":true,
       "remove_admin":true,
       "add_admin":true,
       "list_handles":false
       }
    },
  "data":"B/MAAAAKMC5OQS8xMDkxNgAAAMgAAA==",
  "timestamp":"2012-10-26T05:52:43Z",
  "ttl_type":0,
  "ttl":86400,
  "refs":[],
  "privs":"rwr-"
  }
]
```

HTTP PUT

Now, say we'd like to change the information in this Handle. We could store the following JSON in file `TEST.json`:

```
[
  {
    "type": "URL",
    "parsed_data": "http://www.sara.nl/Nagioscheck_updated"
  },
  {
    "type": "EMAIL",
    "parsed_data": "john.doe@example.com"
 }
]
```

Let's try if we can overwrite Handle `10916/58405ca4-1f31-11e2-ba9d-14feb57d1255` with this new information:

```
curl --user 10916:<$EPIC_PASSWORD> \
     --upload-file TEST.json  \
     --header 'Content-type: application/json' \
https://epic3.storage.surfsara.nl/v2/handles/10916/58405ca4-1f31-11e2-ba9d-14feb57d125
5
```

When invoked with the `--upload-file <filename>` option, curl uploads the file with an `HTTP PUT` request.

The data now has been updated. The type "URL" has been updated. There is a new type "EMAIL". And the old type "directory" has been removed.

```
[
```

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.

```
{
"idx": 1,
"type": "URL",
"parsed_data": "http://www.sara.nl/Nagioscheck_updated",
"data": "aHR0cDovL3d3dy5zYXJhLm5sL05hZ2lvc2NoZWNrX3VwZGF0ZWQ=",
"timestamp": "2012-10-26T08:36:39Z",
"ttl_type": 0,
"ttl": 86400,
"refs": [],
"privs": "rwr-"
},
{
"idx": 2,
"type": "EMAIL",
"parsed_data": "john.doe@example.com",
"data": "am9obi5kb2VAZXhhbXBsZS5jb20=",
"timestamp": "2012-10-26T08:36:39Z",
"ttl_type": 0,
"ttl": 86400,
"refs": [],
"privs": "rwr-"
},
{
"idx": 100,
"type": "HS_ADMIN",
"parsed_data": {
  "adminId": "0.NA/10916",
  "adminIdIndex": 200,
  "perms": {
    "add_handle": true,
    "delete_handle": true,
    "add_naming_auth": false,
    "delete_naming_auth": false,
    "modify_value": true,
    "remove_value": true,
    "add_value": true,
    "read_value": true,
    "modify_admin": true,
    "remove_admin": true,
    "add_admin": true,
    "list_handles": false
    }
  },
"data": "B/MAAAAKMC5OQS8xMDkxNgAAAMgAAA==",
"timestamp": "2012-10-26T05:52:43Z",
"ttl_type": 0,
"ttl": 86400,
"refs": [],
"privs": "rwr-"
}
```

EUDAT receives funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 654065.

]

With the same mechanism, we can also create new PIDs, by simply HTTP PUT-ing our data to the URL of a new, yet to be created, handle:

```
curl --user 10916:<$EPIC_PASSWORD> \
     --upload-file TEST.json  \
     --header 'Content-type: application/json' \
     https://epic3.storage.surfsara.nl/v2/handles/10916/TEST_HANDLE
```

This yields (truncated and abbreviated):

```
...
HTTP/1.1 201 Created
id="rackful_description">A new resource was created:
location "TEST_HANDLE"
...
```

This time, we get an `HTTP/1.1 201 Created`, which tells us that the URL we sent our data to didn't exist yet, but that it has been created by our request (see also the `Location:` response header.

So, as to be expected from a RESTful Web Service, HTTP PUT can be used for both resource *creation* and resource *updates*. It is a common misconception that HTTP's POST, GET, PUT and DELETE verbs correspond one-on-one to Create, Read, Update and Delete actions (a.k.a. CRUD) respectively. This is not the case.

However, there is a way to make your HTTP PUT request "conditional" in the sense that it will only be executed if the URL doesn't exist yet (which constitutes a *create* action), or that it *does* exist yet (which constitutes an *update*). This can be done using HTTP's `If-Match: *` and `If-None-Match *` headers. For example, let's repeat our last resource creation, and this time ask that our request only be handled if the resource doesn't exist yet:

```
curl --user 10916:<$EPIC_PASSWORD> \
     --upload-file TEST.json  \
     --header 'Content-type: application/json' \
     --header 'If-None-Match: *' \
     https://epic3.storage.surfsara.nl/v2/handles/10916/TEST_HANDLE
```

This yields (truncated and abbreviated):

```
... HTTP/1.1 412 Precondition Failed
...
```

Just as specified here in RFC2616, the server comes back with `HTTP/1.1 412 Precondition Failed`, indicating that it couldn't satisfy our precondition that the resource doesn't exist yet.

HTTP POST

You can also POST your data to a suffix URL, instead of PUT-ing it. In this case,The server will create a new suffix for you. Here's an example:

EUDAT receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 654065.

```
curl --user 10916:<$EPIC_PASSWORD> \
     --data-file @TEST.json  \
     --header 'Content-type: application/json' \
     --include \
     https://epic3.storage.surfsara.nl/v2/handles/10916/
```

This yields (truncated and abbreviated):

```
HTTP/1.1 201 Created
Date: Fri, 26 Oct 2012 12:54:38 GMT
Content-Length: 1737
Content-Type: text/html; charset=UTF-8
Location:
https://epic3.storage.surfsara.nl/v2/handles/10916/4d4584e2-1f6c-11e2-bbe9-525400725b8
d
...
...
```

As you can see, an `HTTP/1.1 201 Created` is returned, and as specified by RFC2616, you can find the URL of the newly created resource in the `Location:` response header. In our example, the new resource can be found at https://epic3.storage.surfsara.nl/v2/handles/10916/4d4584e2-1f6c-11e2-bbe9-525400725b8d.

Support

Support for the EPIC API is available via the EUDAT ticketing system through the webform.

If you have comments on this page, please submit them though the EUDAT ticketing system.

Document Data

DocVersion: 1.2

Authors:

Robert Verkerk, robert.verkerk@surfsara.nl

Reviewers:

Kostas Kavoussanakis, kavousan@epcc.ed.ac.uk

Carl Johan Håkansson, cjhak@kth.se

Read more

EUDAT receives funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 654065.