# DWP REST API manual

DWP REST API in ID management system consists of GET, PUT, POST and DELETE methods to retrieve or store data in database. This document serves to describe possible options and attributes in individual calls.

## GenericObject

Attributes:

- **registeredId** of type String
- **type**, possible types are family, user, proband, subject, form, sample, specimen, derivative, aliquot, sample, subsample, rawDataFile, analysis
- **system**, possible systems are CELSPACAdmin, CladeIS, OpenSpecimen, LIMS, Galaxy

## Link

Attributes:

- **left** as UUID of left GenericObject
- **right** as UUID of right GenericObject
- **oriented** of type Boolean, possible values true or false
- **type**, syntax for type is System.type-System.type, for example, CELSPACAdmin.proband-OpenSpecmen.proband

## UserProperty

Attributes:

- **key** as “name” of UserProperty
- **value** as value of UserProperty

## GenericObject methods

### GET with url /generic-objects/

With no attributes retrieves all stored GenericObjects, with selected optional attribues retrieves GenericObjects based on attributes

Optional attributes:

- type
- system
- regiteredId
- from in date format YYYY-MM-DD HH:MM
- to in date format YYYY-MM-DD HH:MM

### POST with url /generic-objects/

Stores GenericObject in to the database

Required attributes:

- registeredId
- type
- system

### PUT with url /generic-objects/{uuid of GenericObject}

Add UserProperty to GenericObject with given UUID

Required attributes:

- key
- value

### DELETE with url /generic-objects/

Set expired attribute to current GenericObject and its Links. With recursive attribute set on true, it sets all GenericObjects and Links on right to be expired.

Required attributes:

- uuid of deleted GenericObject

Optional attributes:

- recursive of type Boolean

## Link methods

### GET with url /links/

Retrieves all sored Links

### GET with url /links/{uuid}

RetrievesLink with specified uuid of Link

### GET with url /links/link

Return Links with given attributes

Required attributes:

- left, UUID of left GenericObject
- right, UUID of right GenericObject

Optional attributes:

- type of Link
- oriented Boolean value about Link orientation

### POST with url /links/

Stores Link in to the database

Required attribute:

- left as UUID of left GenericObject
- right as UUID of right GenericObject
- type as a type of Link
- orietnted as Boolean with values true or false

### PUT with url /links/{uuid}

Add UserProperty to link with given uuid

Required attributes:

- key
- value

### DELETE with url /links/{uuid}

Set expired attribute to Link with given uuid

### GET with url /links/filter/{generic-object-uuid}

Retrieves all Links of GenericObject specified with generic-object-uuid

## Query builder

### GET with url /query-builder/

Search for GenericObject using its Links with given Body of request
Syntax for request body:

```
{
    "from": uuid of starting GenericObject,
    "path": path of links,
    "to": index of final GenericObject,
    "query": query of searching criteria
}
```

**Path** attributes serves to choose direction for travelling between GenericObjects. Key words are left, right and prop. Example of syntax is for example, right:right:prop:right, which means that we travel through three Links on right and we will use UserProperty of second visited GenericObject in query attribute.

**To** attributes serves to choose which GenericObject is final and should be retrieved by query builder. GenericObjects are indexed as gobj0 for starting GenericObject, gobj1 for first visited GenericObject on path etc.

**Query** attribute serves to use some search criteria when travelling. For exampe, “gobj1.registeredId = ‘CLS-123‘ AND prop2.key = ‘name’ AND prop2.value = ‘John’”.