MILESTONE

Project Acronym: OpenUp!

Grant Agreement No: 270890

Project Title: Opening up the Natural History Heritage for Europeana

---

MS02: Sample data service mock-up for data quality services - botany

Revision: 1.5

Authors:

Anna Saltmarsh, Royal Botanic Gardens, Kew (RBGK -16)
Nicola Nicolson, Royal Botanic Gardens, Kew (RBGK -16)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	
C	Confidential, only for members of the consortium and the Commission Services	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Author	Organisation	Description
1.1	05/05/2011	N Nicolson	RBGK	Posted on http://open-up.eu/content/botanical-data-quality-service-candidate
1.2	12/05/2011	N Nicolson	RBGK	Discussed with TMG and agreed
1.3	09/06/2011	A Saltmarsh	RBGK	Formatting
1.4	21/06/2011	N.Nicolson	RBGK	Added concept check details and address of mockup. Reformatted to include these.
1.5	28/06/2011	A.Saltmarsh	RBGK	Corrections added as noted by Work Package Leader. References added. Reformatted.

Statement of Originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Distribution

Recipient	Date	Version	Accepted YES/NO
Gavin Malarky - Work Package Leader	28/06/2011	1.5	Yes
Project Coordinator	28/06/2011	1.5	Yes

Table of Contents

1. Introduction

The botanical data quality service has been subdivided into a names check and a concept check. These are each implemented as a HTTP accessed JSON format webservice, and are available from the following URL: http://data.kew.org/openup-data-quality-service/

2. Nomenclatural check

a. Introduction

The initial candidate service is focused on the nomenclatural check - accepting a name (or names) and returning a response to indicate if the name is found in the nomenclatural data source supporting the service.

We have investigated making this service compliant with the Google Refine[1] (previously known as Freebase Gridworks) reconciliation API. In summary this is an HTTP accessed service with JSON format request / response. The service must support callbacks (JSONP http://en.wikipedia.org/wiki/JSONP)[2]

Note: none of the values are set yet, this work is just to investigate the feasibility of using this request / response format.

b. Metadata

The service must support a **metadata request** - called by requesting the reconcilation service URL with no parameters or just a callback function name parameter.

Request

Sample URL: http://data.kew.org/openup-data-quality-service/name-reconciler

Response

```
{
  "name" : "IPNI name reconciliation service",
  "identifierSpace" : "http://lsid.ipni.org",
  "schemaSpace" : "http://lsid.ipni.org" ,
  "view" : {
    "url" : "http://ipni.org/{{id}}"
  }
}
```

Request (using callbacks)

Sample URL: http://data.kew.org/openup-data-quality-service/name-reconciler?callback=my_function

Response (using callbacks)

```
my_function(
{
  "name" : "IPNI name reconciliation service",
  "identifierSpace" : "http://lsid.ipni.org",
```

```
  "schemaSpace" : "http://lsid.ipni.org"
}
)
```

c. Single query mode

Request

Queries are specified in JSON. A single query can be specified as follows:

```
{
    "query" : "Ocimum americanum",
    "limit" : 3,
    "type" : "name",
}
```

The JSON query is specified as a URL parameter, and the service can respond to either GET or POST.

Response

```
{
    result : [
        {
        id: "urn:lsid:ipni.org:names:452857-1"
        , name: "Lamiaceae Ocimum americanum Jacq."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    , {
        id: "urn:lsid:ipni.org:names:452858-1"
        , name: "Lamiaceae Ocimum americanum Auct. ex Benth."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    , {
        id: "urn:lsid:ipni.org:names:316218-2"
        , name: "Lamiaceae Ocimum americanum L."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    , {
        id: "urn:lsid:ipni.org:names:452856-1"
        , name: "Lamiaceae Ocimum americanum L."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    ]
}
```

As per the metadata request, a callback function name can be specified - this will have the effect of wrapping the JSON response in a function call to the specified function name.

d. Multiple query mode

Here, multiple queries are passed in a single request. The individual queries are specified as above, but are keyed. These keys are used to link the results to the queries.

Request

```
{ "q1" : {
    "query" : "Ocimum americanum",
    "limit" : 3,
    "type" : "name",
}
, "q2" : {
    "query" : "Ocimum africanum",
    "limit" : 3,
    "type" : "name",
} }
```

Response

```
{
    q2 : {
    result : [
    {
        id: "urn:lsid:ipni.org:names:452853-1"
        , name: "Lamiaceae Ocimum africanum Lour."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    ]
    }
    ,
    q1 : {
    result : [
    {
        id: "urn:lsid:ipni.org:names:452857-1"
        , name: "Lamiaceae Ocimum americanum Jacq."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    , {
        id: "urn:lsid:ipni.org:names:452858-1"
        , name: "Lamiaceae Ocimum americanum Auct. ex Benth."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    , {
        id: "urn:lsid:ipni.org:names:452856-1"
        , name: "Lamiaceae Ocimum americanum L."
        , type: ["name"]
        , score: "100"
        , match: "true"
    }
    ]
    }
}
```

As per the metadata and single query requests, a callback function name can be specified.

e. Specifying extra data

The query object can also include a properties array, which is a way to send extra data to the service. This could be used to optionally include fields such as family name and authorship:

```
{
 "query": "Ocimum gratissimum var. gratissiumum",
 "type": "name",
 "type_strict": "should",
 "properties": [ {
  "pid": "family",
  "v": "Lamiaceae"
 }]
}
```

3. Taxonomic check

a. Introduction

This service has much the same interface as the nomenclatural check - accepting a name (or names) and returning a response to indicate the taxonomic status of the name in the underlying concept store, and either a list of synonyms or accepted names.

The **status** flag will be one of A (accepted), S (synonym), U (unplaced).

Note: none of the values are set yet, this work is just to investigate the feasibility of using this request / response format.

b. Metadata

The service must support a **metadata request** - called by requesting the reconcilation service URL with no parameters or just a callback function name parameter.

Request

Sample URL: http://data.kew.org/openup-data-quality-service/concept-reconciler

Response

```
{
  "name" : "Kew concept reconciliation service",
  "identifierSpace" : "http://data.kew.org",
  "schemaSpace" : "http://data.kew.org" ,
}
```

Request (using callbacks)

Sample URL: http://data.kew.org/openup-data-quality-service/concept-reconciler?callback=my_function

Response (using callbacks)

```
my_function(

{
  "name" : "Kew concept reconciliation service",
  "identifierSpace" : "http://data.kew.org",
  "schemaSpace" : "http://data.kew.org" ,
}
```

)

c. Single query mode

Request

Queries are specified in JSON. A single query can be specified as follows:

```
{
    "query" : "Ocimum americanum",
    "limit" : 3,
    "type" : "concept",
}
```

The JSON query is specified as a URL parameter, and the service can respond to either GET or POST.

Response

```
{
        result : [
                {
                id: "12345"
                , name: "Ocimum americanum"
                , type: ["concept"]
                , score: "100"
                , match: "true"
                , status: "A"
                , synonyms: [
                        "Poa annua"
                        , "Poa alpina"
                ]
                }
              ]
}
```

As per the metadata request, a callback function name can be specified - this will have the effect of wrapping the JSON response in a function call to the specified function name.

d. Multiple query mode

Here, multiple queries are passed in a single request. The individual queries are specified as above, but are keyed. These keys are used to link the results to the queries.

Request

```
{ "q1" : {
    "query" : "Ocimum americanum",
    "limit" : 3,
    "type" : "concept",
}
, "q2" : {
    "query" : "Ocimum africanum",
    "limit" : 3,
    "type" : "concept",
} }
```

Response

```
{    q2 :
{
result : [
{
id: "12345"
, name: "Ocimum africanum"
, type: ["concept"]
, score: "100"
, match: "true"
, status: "A"
, synonyms: ["Poa annua" , "Poa alpina" ]
}
]
 }
 ,
q1 : {
result : [
{
id: "12345"
, name: "Ocimum americanum"
, type: ["concept"]
, score: "100"
, match: "true"
, status: "A"
, synonyms: ["Poa annua" , "Poa alpina" ]
}
]
 }
}
```

As per the metadata and single query requests, a callback function name can be specified.

e. Specifying extra data

The query object can also include a properties array, which is a way to send extra data to the service. This could be used to optionally include fields such as family name and authorship, or to supply a name id to which the name has already been matched

```
{
 "query": "Ocimum gratissimum var. gratissiumum",
 "type": "concept",
 "type_strict": "should",
 "properties": [ {
  "pid": "family",
  "v": "Lamiaceae"
 },{
  "pid": "nameid",
  "v": "urn:lsid:ipni.org:names:12345-1"
 }

]
}
```

List of References

1. Google Refine: http://code.google.com/p/google-refine/
2. Wikipedia entry for JSONP: http://en.wikipedia.org/wiki/JSONP