REST API Documentation Documentation

Release 1.0

Indicia Team

Jun 11, 2018

Contents

Contents:

CHAPTER 1

REST API Introduction

This document describes a RESTful API implemented in Indicia and BirdTrack allowing the sharing of records between systems. It can be implemented within other online recording systems wishing to participate in record sharing.

Todo: Appropriate reflection of BSBI and BTO as partners in this development.

BirdTrack and Indicia both collect records from a wide range of taxonomic groups. Although BirdTrack is in its nature primarily a system for recording birds, it has been extended to allow recording of other taxonomic groups such as Odonata, which are frequently included on lists produced by bird recorders. However, the expert verifiers who are engaged with the BTO BirdTrack system do not have the expertise to verify these data and furthermore, these records would be of great interest to recording schemes such as the British Dragonfly Society who are using Indicia and iRecord. Similarly, iRecord is used by recorders to enter records from a wide range of taxonomic groups including birds. However as many bird expert verifiers are already engaged with BirdTrack it would not make sense to encourage them to use iRecord, especially given the fact that the majority of expert verifiers give their time voluntarily. Therefore a mechanism for synchronising records from one online recording system to another and to synchronise verification decisions back again is required.

Note: The RESTful API described here provides the part of the mechanism which exposes records to outside systems; it does not define how those systems should go about pulling the records from another system's API into their database.

In general, the master copy of any record remains in the source system, but other systems will be able to annotate records in the context of verification messages and outcomes. At the minimum, implementations of the REST API provide the following:

- A list of sets of records which are available to the other participating system (projects).
- A list of records entered onto the system since a given date for each project so that they can be synchronised into other systems.
- A list of annotations of records as a result of expert verification. This allows verification outcomes to be exported back to the source of the record. Annotations can be added to a record either by the system performing NBN Record Cleaner style automated checks, by user comments, or by expert verifiers.

Synchronisation requires that the API is implemented by all participating systems. The following steps describe an example configuration for synchronisation between iRecord and BirdTrack:

- BirdTrack declares a project created called "BirdTrack Odonata" which filters BirdTrack records to only the dragonfly and damselfy records.
- BirdTrack is configured to allow iRecord to access it's API and to allow iRecord to access the BirdTrack Odonata project.
- iRecord declares a project created called "iRecord Birds" which filters iRecord records to only the bird records.
- iRecord is configured to allow BirdTrack to access it's API and to allow BirdTrack to access the iRecord Birds project.
- A process is written as part of the BirdTrack system to run periodically (e.g. nightly). This process calls the iRecord RESTful API to retrieve entered or changed records in the iRecord Birds project since the last time the process was run. The records are imported into BirdTrack.
- **A process is written as part of the iRecord system to run periodically (nightly?).** This process calls the BirdTrack RESTful API to retrieve entered or changed records in the BirdTrack Odonata project since the last time the process was run. The records are imported into iRecord.
- The BirdTrack Odonata records are made available to expert verifiers on iRecord. Verification outcomes are stored in the iRecord database as annotations in the `occurrence_comments` table.
- The iRecord Birds records is made available to expert verifiers on BirdTrack. Verification outcomes are stored in the BirdTrack database as annotations.
- The periodic process on BirdTrack requests any verification responses for the BirdTrack Odonata project from the annotations on iRecord and pulls them back into BirdTrack. BirdTrack then notifies recorders as it sees fit.
- The periodic process on iRecord will request any verification responses for the iRecord Birds project from the annotations on BirdTrack and pulls them back into iRecord.
- iRecord then notifies recorders as it sees fit.

The API receives requests via URLs and returns JSON format responses. Although other formats (NBN exchange, CSV, XML etc) could be considered, the existing NBN REST API also returns JSON so this will limit the number of technologies users of these APIs need to learn.

CHAPTER 2

Authentication

Several authentication mechanisms have been reviewed to see if they meet the needs of this API. In particular, the mechanism must be:

- Secure
- RESTful (and therefore stateless)
- Easily understood and implemented.

Login/session based approaches are not stateless and therefore not truly RESTful, so authentication data must be presented with every request.

In order to achieve the requirements a protocol based the standard method of using an HMAC (keyed-hash message authentication code) has been implemented:

1. The requesting entity creates a HMAC-SHA1 value of the complete request url (including parameters). The hash value uses the user password as the shared secret.
2. The requesting entity adds an Authorization header to the request containing the following string USER:[user_id]:HMAC:[hmac] where:
    - [user_id] is the requesting user's agreed system identifier
    - [hmac] is the HMAC-SHA1 value computed in (1)
3. The receiving entity recomputes the HMAC-SHA1 in the same manner as (1) and any authorisation failure is returned as HTTP 401 Unauthorized.

This authentication should provide suitable protection against tampering and sufficient level of authentication providing the shared secret is sufficiently long.

The following example PHP snippet illustrates the code required for authentication against the REST API:

```
<?php
$sharedSecret = 'mypassword';
$userId = 'ME';
$url = 'http://www.example.com/index.php/services/rest/projects';
$session = curl_init();
```

(continues on next page)

(continued from previous page)

```
// Set the POST options.
curl_setopt($session, CURLOPT_URL, $url);
curl_setopt($session, CURLOPT_HEADER, FALSE);
curl_setopt($session, CURLOPT_RETURNTRANSFER, TRUE);
// Create the authentication HMAC.
$hmac = hash_hmac("sha1", $url, $sharedSecret, $raw_output = FALSE);
curl_setopt($session,
    CURLOPT_HTTPHEADER,
    array("Authorization: USER:$userId:HMAC:$hmac")
);
// Do the request.
$response = curl_exec($session);
$httpCode = curl_getinfo($session, CURLINFO_HTTP_CODE);
$curlErrno = curl_errno($session);
// Check for an error, or check if the http response was not OK.
if ($curlErrno || $httpCode != 200) {
  echo "Error occurred accessing $url<br/>";
  echo "Rest API error $httpCode<br/>";
  if ($curlErrno) {
    echo "Error number: $curlErrno<br/>";
    echo 'Error message: ' . curl_error($session) . '<br/>';
  }
  echo 'Response: <pre>' . htmlspecialchars($response) . '<br/>';
  throw new exception('Request to server failed');
}
$data = json_decode($response, TRUE);
echo json_encode($data);
?>
```

CHAPTER 3

Unique record identifiers

Each participating system will be given a unique user ID - a code that unqiuely identifies the system to other participants. The system identifiers can be mutually agreed since a relatively small number of participating systems will exist, for example “BRC” or “BTO” would be suitable candidates.

Presumably each participating system will use a standard relational database model with a primary key for all database tables. As this primary key is likely to be a locally generated sequential integer, the IDs will not be unique across all databases.

Therefore each participating system will need to prefix it’s user_id to its record IDs to make a globally unique ID, so record ID 100 on iRecord might be expressed as BRC100 for example. Limiting these to 3 alphabetical characters makes parsing of IDs easier. The resultant record identifiers only need to be unique within each resource type and not globally across all resource types.

CHAPTER 4

API Versioning

The default version for all calls is the latest available API version on that system. Requests for a specific API version can be made by inserting the API version name into the URL segments, placing it before the resource name. For example:

```
http://example.com/rest/v1.0/projects
```

is equivalent to:

```
http://example.com/rest/projects
```

when the API is at version 1.0.

The API should normally be used without specifying the version and the option to use the version is only recommended in specific circumstances, e.g. during development. This approach ensures that resource URIs are effectively permalinks that will not change over time. For more information on the reasoning here, see http://stackoverflow.com/questions/389169/best-practices-for-api-versioning.

CHAPTER 5

Field formats

Dates in requests and responses are formatted to ISO 8601. The following possibilities are accepted:

Format	Notes
Date	`yyyy-mm-dd`, e.g. 2014-12-25
Date and time	• `yyyy-mm-ddThh:mm:ss`, e.g. 2014-12-25T16:25:27 (without timestamp) • `yyyy-mm-ddThh mm:ss+hh:mm`, e.g. 2014-12-25T16:25:27+02:00 (with timestamp)

CHAPTER 6

API resources

The RESTful API defines a set of resources available at the following URLs and request types. Since it is effectively a uni-directional read only specification, we are using GET for all requests. Resource names are lower case and use hyphens as a word separator, to meet with current best-practice for URI naming.

Because a server might not want to expose all its records to a client, the API includes the notion of projects that a client can access. A project is effectively a filter on the underlying records; the mechanism of how this filter may be defined internally is up to each system. For example, in Indicia a project would use the reporting saved filters system to allow a flexible definition of the list of records available.

6.1 Resource object definitions

The API returns resource objects (in JSON format), either individually, or in lists depending on the API call made. The following list of object types are defined for the API.

6.1.1 project

The metadata describing a set of records on the server which are being made available to a client. A project might, for example, be the bird records from iRecord. For a simplicity of implementation, each project is unique to the calling client (so clients cannot call projects set up for other clients and there is no need for a many-many relationship).

Projects contain the following fields (fields marked with a * are mandatory):

- id* - the unique identifier of the project. This must be provided with requests for taxon-observations and annotations from within this project.
- href* - provides a link back to the resource API endpoint describing the individual project.
- title* - project title.
- description* - a description of the project.

An example project object is:

```
{
  "id":"BTO12",
  "href":"http://www.bto.org/rest/projects/BTO12",
  "title":"BTO Odonata",
  "description":"Odonata records for verification on iRecord"
}
```

6.1.2 taxon-observation

The attributes of a single wildlife record.

The API is based on but not exactly the same as the NBN data exchange format field specifications to define taxon observations, as described in the guide to the NBN data exchange format Version 2.7, September 2014. Field names are always lowercased. The RecordKey field is replaced by an id field to keep taxon-observations consistent with other entities exposed by the API. In addition to the fields defined by the NBN exchange format, a lasteditdate field is required.

Todo: Consider case issues here - should fields be lowercased? T/F values are also inconsistently cased throughout the spec.

Taxon observations contain the following properties. Properties marked with a * are mandatory though check the property description for rules relating to the specific field.

- id* - The unique identifier of the observation
- href* - link to the observation's URI. This can be omitted if the API implementation does not support access to a single observation by ID.
- srchref - link to the URI of the object in its originating location, if different to href.
- datasetName - name of the dataset this record was sourced from.
- taxonVersionKey* - the Taxon Version Key from the UKSI species database.
- taxonName* - the taxon name used by the recorder.
- zeroAbundance - set to T to indicate an absence record or F otherwise. The default if not provided is F.
- count - integer value representing the count.
- delete - set to T to indicate this record has been deleted.
- sensitive - set to T to indicate a sensitive record or F otherwise. The default if not provided is F.
- startDate* - the start of the range of dates that the record covers, which will be the same as the enddate field when a record occurred on a single date.
- endDate* - the end of the range of dates that the record covers, which will be the same as the startdate field when a record occurred on a single date.
- dateType* - see the NBN Gateway Exchange format (http://www.nbn.org.uk/Share-Data/Providing-Data/NBN-Data-Exchange-format.aspx) for a definition of how date types are defined.
- siteKey - a unique ID for the site if available.
- siteName - the name of the site provided with the record.
- gridReference* - the grid reference notation for the record. Mandatory unless east and north are provided. British National Grid or Irish Grid notation depending on projection.

- east* - position of record in east/west direction. Mandatory unless gridreference is provided. Either a decimal longitude or easting.
- north* - position of record in north/south direction. Mandatory unless gridreference is provided. Either a decimal latitude or northing.
- projection* - indiciates the projection used for gridreference, east and north fields. Can be:
  - OSGB
  - OSI
  - WGS84
  - OSGB36
- precision* - the spatial precision of the georeference in metres. Typically the size of the grid square.
- recorder* - the recorder name(s).
- determiner - the name of the person providing the initial identification.
- lastEditDate - returns the date and time time of last edit.

An example taxon-observation object is:

```
{
  "id":"BRC100",
  "href":"http://example.com/rest/taxon-observations/BRC100",
  "srchref":"http://source-server-at-brc.com/rest/taxon-observations/BRC100",
  "datasetName":"iRecord::Mammals::Dorset Mammal Group",
  "taxonVersionKey":"NHMSYS0000530482",
  "taxonName":"Red Kite",
  "startDate":"2014-07-12",
  "endDate":"2014-07-12",
  "dateType":"D",
  "gridReference":"SU956436",
  "projection":"OSGB",
  "precision":"8",
  "recorder":"Joe Brown",
  "lastEditDate":"2014-09-12T13:24:11"
}
```

6.1.3 annotation

The definition of an annotation against a taxon-observation. An annotation is an extra piece of information added after the initial record creation event and may describe a user comment, verification event or redetermination of the record.

Annotations contain the following fields (fields marked with a * are mandatory):

- id* - The unique identifier of the annotation
- href* - link to the annotation's URI. This can be omitted if the API implementation does not support access to a single annotation by ID.
- taxonObservation* - contains a child-object, itself containing the id and href for the taxon observation being annotated
- taxonVersionKey* - the unique identifier of the taxon concept that this annotation was made against. This might differ from the original or current taxon concept associated with the record. This allows annotations to maintain an audit trail of the changing opinions of a record's identification.
- comment - free text

- statusCode1 - either A (accepted), U (unconfirmed) or N (not accepted) to indicate a status if this annotation is setting the verification state of the record.
- statusCode2 - provides additional detail regarding the status code. For accepted records, can be 1 (correct) or 2 (considered correct). For unconfirmed records, can be 3 (plausible) or 4 (not reviewed). For not accepted records, can be 5 (unable to verify) or 6 (incorrect).
- emailAddress - optionally contains the email address of the person creating the annotation. It is recommended that when a user takes an action that results in an annotation (such as commenting on or verifying a record), then the system should give the user an option to opt in to providing their email address. If provided, then on other systems receiving the annotation, the email address must only be made available to the recipient of the notification. This allows an external communication thread to start to discuss the record. Note that email addresses should not be provided if the user creating the annotation has not opted in.
- question - t or f to indicate true or false. If true, then this annotation contains a question that needs answering.
- authorName* - name of the comment author.
- dateTime* - ISO 8601 date format for the timestamp of the annotation.
- lastEditDate - returns the date and time time of last edit.

An example annotation object is:


```
{
  "id":"BRC452",
  "href":"http://indicia.org.uk/rest/annotations/BRC452",
  "taxonObservation":{
    "id":"BRC251",
    "href":"http://indicia.org.uk/rest/taxon-observations/BRC251"
  },
  "taxonVersionKey":"NBNSYS0012345678",
  "comment":"Some text commenting on the record",
  "statusCode1":"A",
  "statusCode2":"1",
  "emailAddress":"example@example.com",
  "question":"f",
  "authorName":"John Smith",
  "dateTime":"2014-02-01T09:00:22+05:00",
  "lastEditDate":"2014-02-01T09:00:22+05:00"
}
```

CHAPTER 7

GET responses

Responses to HTTP GET requests are either a single object (one of _project_, _taxon-**observation_** or _annotation_ as described above) or a list/array of objects. When returning a list of objects, the each individual response includes a single page of objects and it may be necessary to make multiple calls to page through the dataset. Therefore the structure also includes metadata to simple support pagination by providing links to the current, next and previous page. The following template is used:

```
{
  "data":[
    { project, taxon-observation or annotation object },
    { project, taxon-observation or annotation object },
    { project, taxon-observation or annotation object },
    etc
  ],
  "paging":{
    "self":"uri for current page in set",
    "previous":"uri for previous page in set",
    "next":"uri for next page in set",
  }
}
```

The next and previous page links are only provided when there is a next or previous page available in the dataset.

7.1 Resource API end-points

The following list of end-points are exposed by an implementation of the REST API:

7.1.1 GET /projects

Retrieves a list of projects available to the client.

The server side needs a mechanism for associating records to "projects" and for associating projects to the accessing client's authorisation. So, iRecord might be able to access BirdTrack Odonata records but not bird records, therefore

BirdTrack will need to be able to identify the Odonata records with a unique project ID and to recognise that iRecord can access this project.

Note: In iRecord, it is likely that projects will be managed using the existing filters system, giving great flexibility over the records exposed. This is a detail of implementation which does not affect the transfer specification.

Request fields

- page_size - number of records to return in the page.
- page - index of the page to return, default 1.

Response status codes

200 - Success 401 - unauthorized

Response

A successful request receives a list of projects in JSON format, using the *GET response template* and the *project resource format*.

7.1.2 GET /taxon-observations

Retrieve a list of records as a JSON array.

By default the request returns 1 day of records if no end date is specified.

If there are no records, then an empty array should be returned.

Implementations of this API might choose to reject requests for date ranges wider than 1 week, but this restriction can be omitted where the API is being put to wider use.

Deleted records can be included in batches of updates. A deleted record will at the minimum include the unique identifier for the record plus a flag "Deleted=t".

Request fields

Fields marked with a * are mandatory so must be included in the request.

- proj_id* - ID of the project whose records are being requested.
- edited_date_from* - format yyyy-mm-dd or ISO 8601 yyyy-mm-ddThh:mm:ss. Limits to records created or updated on or after this date.
- edited_date_to - format yyyy-mm-dd or ISO 8601 yyyy-mm-ddThh:mm:ss. Limits to records created or updated on or before this date.
- page_size - number of records to return in the page.
- page - index of the page to return, default 1.

Note that this date format confirms to ISO 8601.

Response status codes

- 200 - Success
- 400 - Bad request (Invalid parameters)
- 401 - unauthorized

Response

A successful request receives a list of taxon-observations in JSON format, using the *GET response template* and the *taxon-observation resource format*.

Note: The server is responsible for ensuring that the default sort order of any taxon observations returned is stable and not affected by edits happening whilst the client pages through the dataset. For example, a sort by creation timestamp or record ID (if sequentially generated) would be appropriate.

7.1.3 GET /annotations

Retrieve a list of annotations as a JSON array.

Use query parameters in the URL to filter – e.g. edited_date_from, edited_date_to to define the date range for edits to include. If there are no results then an empty array is returned.

Request fields

Fields marked with a * are mandatory.

- proj_id* - the ID of the project whose annotations are being requested.
- edited_date_from* - format yyyy-mm-dd or ISO 8601 yyyy-mm-ddThh:mm:ss. Limits to records created or updated on or after this date.
- edited_date_to - format yyyy-mm-dd or ISO 8601 yyyy-mm-ddThh:mm:ss. Limits to records created or updated on or before this date.
- page_size - number of records to return in the page.
- page - index of the page to return, default 1.

Note that the 2 date filter fields relate to the edit date of the annotation record itself and are independent of the taxon-observation's edit date.

Response status codes

- 200 - Success
- 400 - Bad request (Invalid parameters)
- 401 - unauthorized

Response

A successful request receives a list of annotations in JSON format, using the *GET response template* and the *annotation resource format*.