FIMS documentation Documentation

Release 1.0

John Deck, RJ Ewing

Apr 27, 2018

About the FIMS System

All source code mentioned in this documentation is open source and freely available and can be found in appropriate repositories living under the Biocode, LLC GitHub Organization.

CHAPTER 1

Introduction

Biocode-FIMS is used for data validation, expedition planning, and data management for field-based surveys enabling tracking physical objects including organisms, soil cores, water samples, and sub-samples. If you would like to start your own Biocode FIMS project, you can either download and install the relevant modules (all freely available) or contact the owner of the BiSciCol FIMS installation code site to see if you can be added as a project to this installation.

CHAPTER 2

Biocode FIMS Web Application

Biocode FIMS has multiple running instances, in support of a variety of field-based sampling projects. An example of how the FIMS components work together is running at the BiSciCol homepage. The biscicol-fims codebase is one example of a FIMS implementation which features data validation, a graph-based data storage engine, assignment of globally unique identifiers (ARKS) for expeditions, datasets, and all samples and processes. Project administrators control and validate specific fields, while users can choose to add their own data to spreadsheet templates.

For developers: All Biocode FIMS installations use the biocode-fims-commons code repository and may also include the biocode-fims-fuseki codebase for working with triples and/or the biocode-fims-sequences codebase for working with fasta and fastq sequencing files.

2.1 How FIMS is organized

The following sections describe how data is stored in the FIMS system. See the identifiers section for more information how identifiers are applied at each level.

2.1.1 Projects

A project defines a set of data held in common by a group of contributors. The data from a single project shares common concepts, attributes, and validation rules. It is owned by a project administrator user who has the power to set validation rules, concept mappings, and users that are permitted to share data under the project. New projects are created infrequently and require super-user status to create (currently the owner of BCID site itself). However, once created, project values can be edited by an assigned project administrator.

To create a new project, please contact the BCID owner.

2.1.2 Expeditions

Projects contain many expeditions. An Expedition refers to a group of datasets and is associated with a project. Users may freely create expeditions.

2.1.3 Datasets

Expeditions contain many datasets. A dataset typically refers to an excel spreadsheet. When loaded through the associated Biocode-FIMS expeditions, a reference to this dataset is stored as part of the Biocode-FIMS BCID system with a unique ARK. When a dataset is created, it is given a unique code, which is owned by the user that first uploaded it. This user may choose, to re-load this dataset again, in which case a new reference will be loaded here.

2.1.4 Resources

All expeditions contain one or more resources. A special resource identifier is created for each of the expedition resources. For example, a project cataloging tissue samples may choose to define a resource for the tissue itself, a resource for the collecting event data associated with the tissue, and an identification resource containing information about the taxonomic name associated with the tissue. Or, a project may choose to simply bundle all of this information into a single resource describing all of this data.

CHAPTER 3

Biocode FIMS ElasticSearch Query Information

Below you will find information about how queries are matched for Biocode FIMS implementations using ElasticSearch.

3.1 Querying

Currently queries can be filtered on:

- expeditions
- columns
- full text search

3.1.1 columns

This is a `{key}:{value}` pair, where `{key}` is either a `columnName` or a `columnUri` defined in the project configuration file (*config files*) or *_all*. When using `{key}:{value}` queries, the `{value}` is an exact match.

3.1.2 _all query

The _all Query is a special query which performs a full text search across all columns in the project. This is a much more lenient query compared to the exact match detailed above and follows the tokenization process detailed *below*.

3.2 Tokenization

Currently the tokenization process splits words on any non-letter character or camelCase. The tokens are then lowercased and further processed to remove any possessives and plurals.

Tokenization Ex:

```
"manyDonkeys" -> ["many", "donkey"]
```

The above result match the following queries:

```
_all=many

_all=manyMore

_all=many Donkeys

_all=manyDonkeys

_all=DONKEYS

_all=donkey
```

However it will not match:

```
_all=manydonkeys
```

For more information, you can view the Elastic Search Analysis Docs. Our current anlyzer settings can be found *here*.

CHAPTER 4

bioValidator: A desktop validation tool (legacy)

4.1 About

bioValidator is a desktop appliction that is "maintenance" only mode and is included here since it is closely related to other FIMS projects and some projects still use bioValidator. Included in bioValidator is a photo-matcher, which is a visual method for matching photos with spreadsheet elements and renames photos according to Collector's Specimen Numbers.

4.2 Getting Started

Download latest release. bioValidator installs and runs in a single file. It is reccomended, however, that you create its own directory when downloading it. If you have Java 1.6 or later installed on your computer, everything will run fine. Otherwise, you will need to download and install Java 1.6

Entering Data For Biocode by using the Biocode Template and view instructions for entering data

4.3 Building rules for data validation

Data validation is based on a set of rules that are defined in an XML file. Each rule can be set to a level of "Error", which means that no downstream processing or PhotoMatching can occur, or "Warning" which means that this is an issue that can be fixed later and not critical for fixing immediately. Tools for building validation XML files are forthcoming in later releases. Meanwhile, it is important to know that all XML rules files themselves are validated against an <a href="http://biocode.berkeley.edu/bioValidator">XSD schema document</a>

4.4 Validating Data

You must first download the appropriate Excel Template file to begin entering data. The excel template and instructions for filling it out are available on the <a href="http://biocode.berkeley.edu/batch_upload_help.html">Biocode

Website</a>. Once you have entered some data, you can validate it by opening bioValidator and clicking on “Load Spreadsheet”. You can use the tools on the screen to view errors and/or warnings. Once you fix any errors or warning, save your spreadsheet and re-load it by clicking on “Load Spreadsheet” again.

4.5 PhotoMatching

In order to photomatch, you must first have a spreadsheet that has passed validation with no errors (though you may proceed with warnings). Click on the “Input Directory” to choose a directory that contains your photos. bioValidator will not modify photos in this directory, but it will create a subdirectory called “.bvthumbs” which the application uses to display thumbnails. Once you have loaded a directory, choose an output directory by clicking on “Output Directory”. This directory will store photos that have been renamed during the matching processes. Create matches of photos to specimens by scrolling through each and pressing Control & the mouse click button at the same time. You will see photos copied to each specimen.

Note that if you have an output directory with appropriately named photos that have already gone through the photomatching processing, they will appear under the specimen information.

CHAPTER 5

Identifiers

FIMS uses a centralized minting service to assign identifiers for three types of identifiers: expeditions, datasets, and resources. The three types of identifiers are described below.

Each FIMS system installation must use its own name assigning authority number and register with California Digital Library's EZID service to mint Archival Resource Keys (ARKs).

5.1 Expedition identifiers

- resourceType: http://purl.org/dc/dcmitype/Collection
- Mutable, representing the most current version of a particular spreadsheet
- **Metadata:**
    - expeditionCode
    - expeditionTitle
    - userId (who created this expedition)
    - ts (when loaded)
    - projectId (project this belongs to)
    - public (public or not)

5.2 Dataset identifiers

- resourceType: http://purl.org/dc/dcmitype/Dataset
- Immutable
- Belongs to a specific expedition
- **Metadata:**

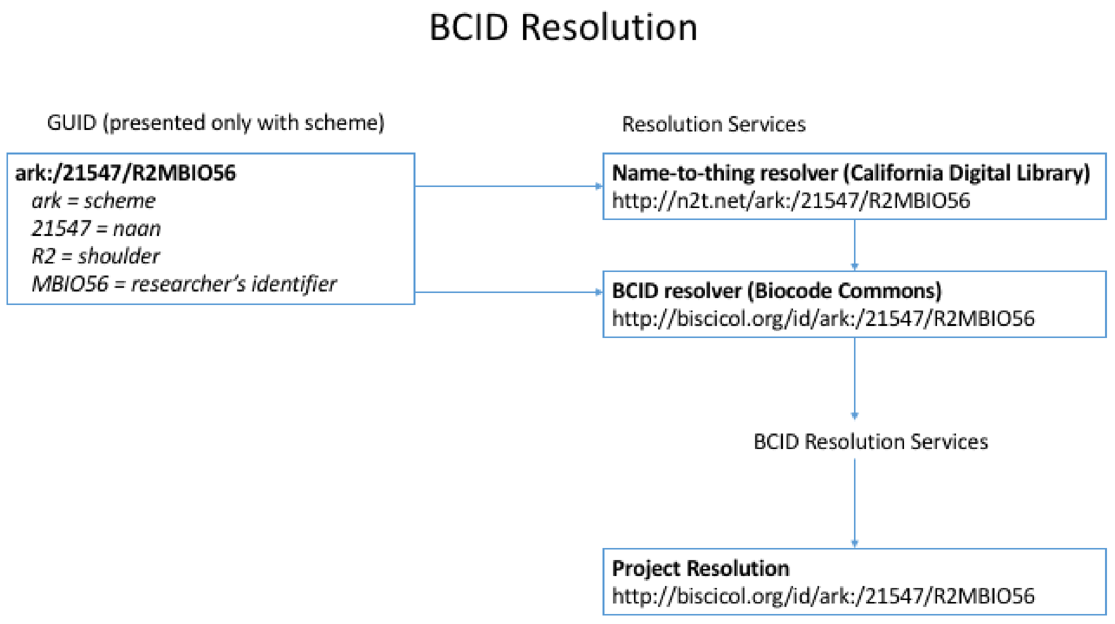
  - webAddress (where this dataset can be found, in its native format, depending on installation)
  - userId (who uploaded this dataset)
  - doi (an optional doi, in addition to the created ARK)

5.3 Resource identifiers

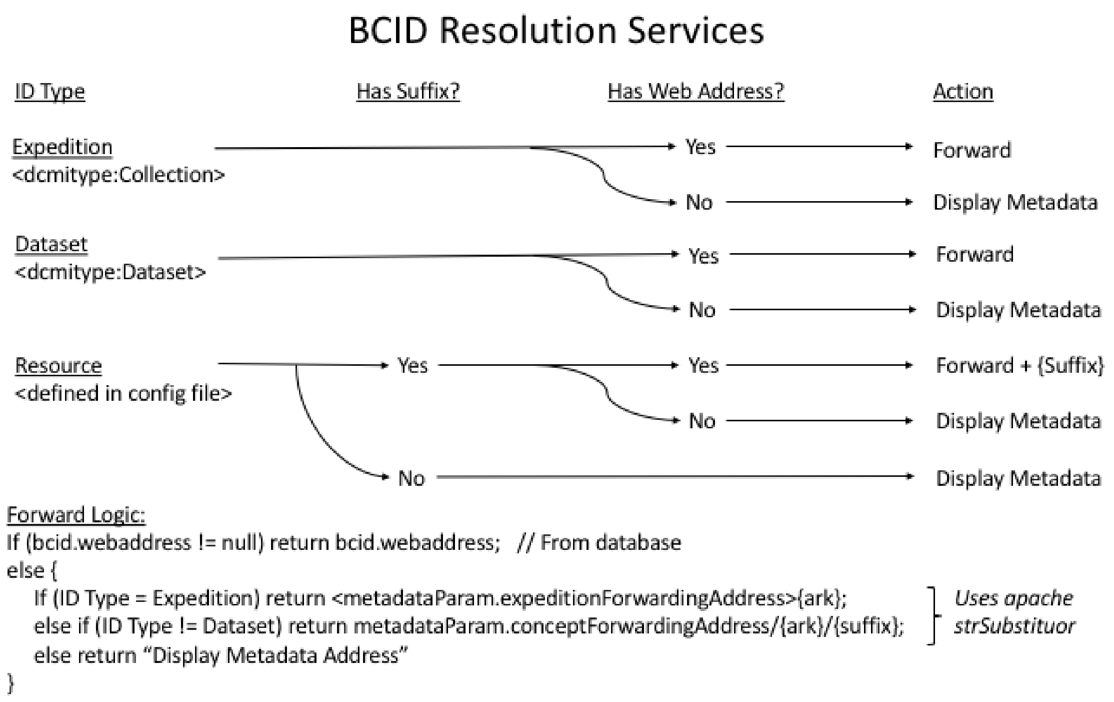
- resourceType: defined in configuration file
- Belongs to an expedition. Multiple resources may be specified for each expedition.
- Implements suffix-passthrough feature to identify individual resources within each dataset. For example, a single "Material Sample" identifier is created for each expedition. If the expedition has 1000 rows representing physical samples, 1000 identifiers can be resolved by appending a locally unique suffix on to the Resource Identifier root.
- A resource identifier plus the locally unique primary key loaded for the most recent dataset in an expedition forms the globally unique identifier for a particular resource.

5.4 BCID Resolution System

The following illustration shows how BCIDs work with local identifiers, the world wide web, and EZID's name-to-thing resolution service. A field researcher uses their own numbering system (e.g. 'MBIO56'), and uploads their data to FIMS, which assigns it to a resource category (e.g. 'R2'). The FIMS system itself is registered under the ark: scheme, and has a name assigning authority number (NAAN) of 21547. Resolution requests coming through name-to-thing are re-directed to the BCID resolution service.


BCID Resolution
GUID (presented only with scheme)
ark:/21547/R2MBIO56
ark = scheme
21547 = naan
R2 = shoulder
MBIO56 = researcher's identifier
Resolution Services
Name-to-thing resolver (California Digital Library)
http://n2t.net/ark:/21547/R2MBIO56
BCID resolver (Biocode Commons)
http://biscicol.org/id/ark:/21547/R2MBIO56
BCID Resolution Services
Project Resolution
http://biscicol.org/id/ark:/21547/R2MBIO56

The following chart shows how BCID resolution works for expeditions, datasets, and resources in the FIMS system with actions falling under forwarding, or metadata display. Forwarding behaviour is determined by either the specification of a target webaddress in the database, or absent that, a specification in the project's configuration file.



Forward Logic:

```
If (bcid.webaddress != null) return bcid.webaddress;   // From database
else {
    If (ID Type = Expedition) return <metadataParam.expeditionForwardingAddress>{ark};
    else if (ID Type != Dataset) return metadataParam.conceptForwardingAddress/{ark}/{suffix};
    else return "Display Metadata Address"
}
```

NOTE: This page needs Updating!

CHAPTER 6

Introduction to BCIDs

BCIDs addresses offer a robust, easy to use service, extending the [http://n2t.net/ezid California Digital Library EZID] solution for the biological collections community. EZIDs offer ease of use, scalability, long-term persistence, resolvability, and a consistent metadata format. BCIDs extend this by offering an easy mechanism to assign billions of identifiers right now for data elements (individual samples or processes acting on those samples). BCIDs allow any local ID/UUID to resolve to a group level identifier. Organizations have the option of later purchasing their own EZID subscription for more control over their identifiers. All of the representations that are referred to by BCIDs fall under the [http://creativecommons.org/licenses/by/3.0/ Creative Commons Attribution 3.0 License], enabling others to reference your identifiers and the objects that they refer to as long as they attribute them. Since BCIDs are self-describing in HTML and RDF/XML, downstream users need only maintain the original identifier

6.1 Discussion

BCIDs are hierarchical identifiers, allowing a single "group" to represent and resolve many sub-identifiers. A group is associated with a column of local identifiers, such as a set of specimens, sharing a single common concept. These concepts are, for now, a [http://biscicol.org/bcid/concepts.jsp constrained list] defined by the biodiversity informatics communities and assembled by the [http://biscicol.org BiSciCol project]. Elements are references to individual instances, such as an individual specimen.

A data group may have the following form:

- ark:/21547/R2

A data element within the group may extend this in the following manner:

- ark:/21547/R2MBIO56

The structure above defines the following segments:

||description||value||notes|| ||scheme||ark:||all BCIDs are ARKs|| ||NAAN (Name Assigning Authority Number)||21547||all BCIDs use this NAAN|| ||data group||R2||The data group identifier must be assigned by BCID|| ||data element||MBIO56||The user can supply a local ID, Darwin Core Triplet, or UUID||

Data elements share a root structure with a data group. The delimiter between the scheme, NAAN, and data group is a forward slash. Data group identifiers always end with the first number and data elements immediately follow this number.

Data groups are technically not datasets. This is because a data group must be defined by a single conceptual entity, and stands for that conceptual entity for all child elements. For instance, we can define a data group of "Sound". Any unregistered element that appears following the data group designation will technically be a type of "Sound". This is useful, for instance, if we have scores of objects or processes that have no web-resolution but we still want to be able to refer to them uniquely and know they are an instance of "Sound".

We can also attribute a dataset to one or more data groups. The dataset can be an ARK or a DOI (issued separately).

6.2 Resolution w/ Suffixes

N2T resolver = {{{http://n2t.net/{ARK}}}} (e.g. http://n2t.net/ark:/21547/R2)

N2T metadata = {{{http://n2t.net/ezid/id/{ARK}}}} (e.g. http://n2t.net/ezid/id/ark:/21547/R2)

BCID resolver = {{{http://biscicol.org/id/{ARK}}}} (e.g. http://biscicol.org/id/ark:/21547/R2)

BCID Metadata = {{{http://biscicol.org/id/metadata/{ARK}}}} (e.g. http://biscicol.org/id/metadata/ark:/21547/R2)

BCID Suffix resolver = {{{http://biscicol.org/id/{ARK}_{Suffix}}}} (e.g. http://biscicol.org/id/ark:/21547/R2MBIO56)

BCID Suffix Metadata = {{{http://biscicol.org/id/metadata/{ARK}_{Suffix}}}} (e.g. http://biscicol.org/id/metadata/ark:/21547/R2MBIO56)

Data groups can be assigned a target URL which indicates where to resolve this group. If the "Maintain local ID's" box is checked on data group creation then that indicates suffixes will be passed through to the resolution target. The following table illustrates how identifiers resolve. Note that the N2T resolver does not handle suffix-passthrough *yet* so in the short term, N2T will only resolve data groups without suffixes.

||BCID||Maintain Local ID's||Target URL (group or element)||Ultimate Resolution Target|| ||http://n2t.net/ark:/21547/R2||true||NULL||(BCID metadata)|| ||http://n2t.net/ark:/21547/R2||false||NULL||(BCID metadata)|| ||http://n2t.net/ark:/21547/R2||true||http://biocode.berkeley.edu/specimens/||(BCID metadata)|| ||http://n2t.net/ark:/21547/R2||false||http://biocode.berkeley.edu/specimens/||http://biocode.berkeley.edu/specimens/|| ||http://n2t.net/ark:/21547/R2MBIO56||true||http://biocode.berkeley.edu/specimens/||http://biocode.berkeley.edu/specimens/MBIO56|| ||http://n2t.net/ark:/21547/R2MBIO56||false||http://biocode.berkeley.edu/specimens/||http://biocode.berkeley.edu/specimens/||

6.3 Relationship with EZIDs

All group level identifiers are registered as EZIDs. It is possible to also register element level identifiers as EZIDs. Please talk to us if you would like to do this.

6.4 RDF/XML Accept Headers

Calling the Metadata or Resolution targets using the BCID resolver will return RDF/XML. To enable this, you must pass in the "application/rdf+xml" accept header. To do this in curl, try:

{{{curl -H "Accept: application/rdf+xml" http://biscicol.org/id/metadata/ark:/21547/R2MBIO56}}}

You can also try it out using [http://linkeddata.informatik.hu-berlin.de/uridbg/index.php?url=http%3A%2F%2Fbiscicol.org%2Fbcid%2Fid%2Fark%3A%2F21547%2FR2&useragentheader=&acceptheader=application%2Frdf%2Bxml an online tool using and the "ark:/21547/R2" BCID].

6.5 Developer Information

- View the list of [http://biscicol.org/id/bcid.wadl BCID REST services]
- [http://biscicol.org/bcid/javadoc/index.html JavaDocs] are available
- This is an open source project, please contact the owner to contribute!

CHAPTER 7

Samples and Sub-sampling

NOTE: this section assumes the fuseki data storage engine is used in your FIMS installation.

FIMS is primarily a system of samples which are the result of some specimen collection process. Each specimen collection process can be uniquely defined by the procedure used, who performed it, when it was performed and subject and target samples. Samples can be thought of in terms of a directed graph with each sample as nodes and processes represented by edges, connecting samples with all respective sub-samples. It is important to maintain all points in the chain where significant transformation of material occurs so we can understand the significance of a distinct sample at any point in the chain. **The FIMS configuration file defines the relationships between samples and processes that join them, mapping a single flat file to a graph-based representation (using the fuseki module)**

For more information about connecting samples and processes, read the publication *Semantics in Support of Biodiversity Knowledge Discovery*.

CHAPTER 8

Installation

This content is for people wishing to install Biocode-FIMS on their own server.

8.1 Details

Biocode-FIMS consists of a core set of Java classes and REST services. Developers have a choice of interacting with the REST services running on the BiSciCol FIMS instance, which has built in EZID minting capabilities, or running their own instance of biocode-fims-commons and installing their own EZID instance requiring a purchase of an EZID account.

To run an instance of FIMS you will need the following components:

- A unix-based server * A java servlet container e.g. Tomcat, Glassfish, Jetty * Connection to a BCID service

8.2 Installation and Build

- Source code is available on this site via Subversion
- Building is done via an Ant build file (provided as part of the distribution)
- a properties file needs to be configured by copying biocode-fims.template to biocode-fims.props (in the root directory of the distribution)

8.3 Optional Component

- A triple-store database connection for storing datasets as RDF triples. We have tested using [http://jena.apache.org/documentation/serving_data/ Apache Fuseki]
- An ElasticSearch instance for indexing. See *ElasticSearch Configuration* for configuration details.

8.4 Additional Installation Instructions

- **The source contains a sample property file called biocode-fims.template which should be copied to biocode-fims.props – yo**
  - Note that references to URLs should begin with "www" as in www.biscicol.org instead of biscicol.org since some browsers (e.g. Firefox) automatically add the "www" and this changes the session designation and creates problems during logging in.

8.5 Notes

- It is recommended ALL service calls in the properties file begin with a "www" in the hostname.

CHAPTER 9

ElasticSearch Configuration

Following is information about configurations changes that are made when setting up a new ElasticSearch cluster

CHAPTER 10

Dynamic Templates

Creating dynamic templates allows us to modify the default index settings and mappings.

10.1 Analyzer

We are currently using a custom default analyzer. To set this up, make a `POST` request to `/_template/{template_name}` with the following in the body:

```
{
  "template" : "*",
  "settings" : {
    "index" : {
      "analysis" : {
        "analyzer" : {
          "default" : {
            "filter" : [
              "word_delimiter",
              "lowercase",
              "asciifolding",
              "kstem",
              "stop"
            ],
            "char_filter" : [
              "html_strip"
            ],
            "type" : "custom",
            "tokenizer" : "standard"
          }
        }
      }
    }
  }
}
```

CHAPTER 11

Index Aliases

Indexes are created for each project. When creating an index, the format is `{projectId}_{todays_date}`. Then we use index aliases to point to the latest index version. The index alias format is `{projectId}`. The advantage to this is that if we need to update any settings, mappings, etc we are able to reindex to existing data and point the index alias to the updated index, without changing the index name.

CHAPTER 12

Implementations

Refer to documentation for particular projects that utilize the FIMS system for instructions on using the interface. This documentation focuses on development for back-end components and more generally on topics that all implementations share, such as minting globally unique identifiers. Here we describe just the BiSciCol implementation to show how one possible FIMS implementation may be used.

12.1 BiSciCol

The BiSciCol implementation uses a triplestore data storage engine, a dedicated EZID name assigning authority number, and supports a number of different projects that use this implementation directly:

- Barcode of Wildlife sites (currently Kenya, Mexico, Nigeria, South Africa)
- University and Jepson Herbarium
- Diversity of the IndoPacific (DIPNet)
- New York Botanical Garden

12.1.1 Instances calling some portion of BiSciCol services but running their own interfaces:

- Amphibian Disease Portal (in development: this instance will only mint expedition identifiers and use validation services)
- Automated Reef Monitoring System Project (in development: substitutes a mysql data storage engine for the triplestore data storage)

CHAPTER 13

XML Configuration File

Each project contains its own xml configuration file. This is where the *Projects* specific configuration is specified. This includes *Resources*, attributes, validation rules, and relations.

13.1 Attributes

13.1.1 DataType

Each attribute may specify a `dataType`. A dataType can be specified to provide additional validation, and in the case of date, datetime, and time, can be used for data formatting. This is especially helpful for standardizing the data to aid in querying and analysis.

The following `dataType` are supported:

- String (default if not specified)
- Integer
- Float
- Date
  - must specify `dataformat` as well
- Time
  - must specify `dataformat` as well
- Datetime
  - must specify `dataformat` as well

CHAPTER 14

REST Services

FIMS REST Services are available at: http://biscicol.org/apidocs/v1.1

14.1 Versioning

FIMS REST Services are now versioned. v1 is the default version. You may specify the version by including the header:

`Api-Version: {version}`

or via the url:

`http://biscicol.org/biocode-fims/rest/{version}/...`

We currently support the following versions:

- v1
- v1.1

more info about the specific version resources to come…

CHAPTER 15

FIMS Commons Javadocs

Javadocs for FIMS Commons code is available at: http://biocodellc.github.io/biocode-fims-commons/

CHAPTER 16

User Accounts

User accounts are not required to lookup/resolve BCIDs. However, they are required to work with projects, expeditions, or create new BCIDs. Here we describe how to obtain a user account for Biocode-

16.1 Account Creation

User accounts can be created by either by the Biocode-Fims instance owner or by project administrators. Project administrators can add any existing user in the Biocode-Fims system as an authorized expedition creator. Talk to your project administrator to be added to a particular project.

[https://github.com/biocodellc/biocode-fims-commons/wiki/OAuth2 Information about Open Authorization]

16.2 Project Administrators

Project administrators are set by the Biocode-Fims instance owner upon request. There is only one designated project administrator per project. The project administrator can add, create, and remove users, set the location of the validation XML file, and define the project abstract.

CHAPTER 17

Minting IDs, Creating Expeditions, and Validating Data using REST calls

The examples presented here demonstrate possible methods for interacting with the FIMS REST services for creating identifiers, expeditions, and validating datasets. The full set of REST service calls are at http://biscicol.org/biocode-fims/rest/fims.wadl . Unless otherwise noted, the parameters in the following examples must be sent with the type 'application/x-www-form-urlencoded'.

17.1 Logging in

All BCID minting functions require you to first login. You can login by sending a POST request to:

```
http://www.biscicol.org/biocode-fims/rest/authenticationService/login

POST Parameters:
    username={yourUsername}
    password={yourPassword}
```

Save the cookies that are returned. How you do this depends on your application (some curl examples: http://fims.readthedocs.org/en/latest/curl_examples.html). Pass the cookies file to the POST requests below to authenticate each request.

17.2 Mint Bcid

Mint a Bcid by sending at minimum a title, webAddress, and resourceType. Send a POST request to:

```
 http://www.biscicol.org/biocode-fims/rest/bcids

 POST Parameters:
    webAddress={dataset_webAddress}
    title={dataset_title}
    resourceType={resourceType}  (e.g. "http://purl.org/dc/dcmitype/Dataset")
```

(continues on next page)

(continued from previous page)

```
    *doi={doi}
    *graph={graph} (A sparql endpoint or other location to be included in sparql␣
↪queries)
    *suffixPassThrough={true|pase} (whether this supports suffixPassthrough}
    *finalCopy={true|false} (if this is a final copy)
* Optional parameters

Send Cookies with login data
```

17.3 Optional minting services

Determine your projectID

Knowing your projectID is useful when working with the minting services presented below. Your username must be added to the projects before using the projects themselves. Talk to your project administrator to have your username added to a project. The following URL lists all available projects in the FIMS system:

```
http://www.biscicol.org/biocode-fims/rest/projects/list
```

Create new Expedition

Expeditions are used in the FIMS system for organizing content that is related to versions of and resources related to, a spreadsheet. Mint an Expedition by sending the following POST request:

```
http://www.biscicol.org/biocode-fims/rest/expeditions

POST Parameters:
    expeditionCode={new_expeditionCode}
    expeditionTitle={new_expeditionTitle}
    projectId={your_projectId}
    public={public_expedition}
    webaddress={target URL for expedition, forwarded when the expedition ID resolved}

Send Cookies with login data
```

Associate Bcid with Expedition by sending POST request

Associate a BCID with an expedition:

```
http://www.biscicol.org/biocode-fims/rest/expeditions/associate

POST Parameters:
    expeditionCode={dataset_expeditionCode}
    bcid={identifier returned in step 2}
    projectId={your_projectId}

Send Cookies with login data
```

Validate Dataset

To validate your dataset, send a POST request to:

```
http://www.biscicol.org/biocode-fims/rest/validate

Send request as type 'multipart/form-data' (in Curl, use -F for form data vs -d)
```

(continues on next page)

(continued from previous page)

```
POST Parameters:
    projectId={your_projectId}
    expeditionCode={your_expeditionCode}
    dataset={your_dataset}

Send Cookies with login data
```

The response is returned as JSON, which will look something like:

```
{"done": [{
    "Samples": {
        "errors": [],
        "warnings": [{
            "Missing column(s)": [
                "yearCollected has a missing cell value",
                "permitInformation has a missing cell value",
                "locality has a missing cell value"
            ]
        }]
    }
}]}
```

CHAPTER 18

curl Examples

NOTE: this page needs updating to show FIMS v2 service examples!

This page explains how to call Biocode-Fims services using curl – For a list of Biocode-FIms service call options see the [http://biscicol.org/biocode-fims/rest/biocode-fims.wadl Service Descriptions].

18.1 Public Content

Attempt a simple call to resolve an ARK (return HTML):

```
curl -L http://biscicol.org/id/ark:/21547/R2
```

Resolve an ARK, returning RDF/XML, for a group + suffix:

```
curl -L  -H "Accept: application/rdf+xml" http://biscicol.org/id/ark:/21547/R2MBIO56
```

18.2 Secure Content

First thing is to authenticate and set cookies. Make sure to do this in a directory where you have write access:

```
curl --data "username=USER&password=PSWD" http://biscicol.org/biocode-fims/rest/
↪authenticationService/login --location --cookie-jar cookies.txt
```

Then, call the service in question, for example, to list all Identifiers associated with the account you authenticated as:

```
curl http://biscicol.org/biocode-fims/rest/bcids/list --cookie cookies.txt
```

If this is a new account without any datasets associated with it, if this request is successful, you will see the result:

```
{"0":"Create new group"}]
```

18.3 Using the Creator

The following is an example of how to create a new data group and register local IDs with BCID. There are two main ways to create new data groups. The first way is to specify the resourceType in the request, such as:

```
curl -d  "title=This is a test&resourceType=dwc:MaterialSample&suffixPassThrough=true
↪" http://biscicol.org/biocode-fims/rest/bcids/ --cookie cookies.txt
```

The second way is to first lookup one of the standard resourceTypes in a list, and then using an integer to call the service:

```
curl http://biscicol.org/biocode-fims/rest/resourceTypes
curl -d  "title=This is a test&resourceTypesMinusDataset=3&suffixPassThrough=true"␣
↪http://biscicol.org/biocode-fims/rest/bcids/ --cookie cookies.txt
```

CHAPTER 19

oauth2

All developers need to register their app. Please contact the system admin to register. You will be issued a client_id and client_secret. The client_secret should be kept private.

19.1 Authorization

Client app will make a GET request to /id/authenticationService/oauth/authorize. This request will contain the following query

- client_id (Required) - The client_id your app was issued during when registered.
- redirect_uri (Required) - The absolute URI you would like the response directed to.
- state (Optional) - Will be returned, unmodified, in the response.

The response will contain the following query parameters:

- code - The random 20 character string used to exchange for an access_token. This code expires in 10 mins and can only be used 1 time.
- state - Only if this parameter was included in the request.

19.2 Access Token

Client app will make a POST request to /id/authenticationService/oauth/access_token. This request will contain the following pa

- client_id (Required) - The client_id your app was issued during when registered.
- client_secret (Required) - The client_secret your app was issued during when registered.
- code (Required) - The authorization code received in the authorization request.
- redirect_uri (Required) - The absolute URI you would like the response directed to. Must be identical to the redirect_uri provided in the authorization request.

- state (Optional) - Will be returned, unmodified, in the response.
- grant_type (Optional) - If grant_type is "password", and a username and password is provided, the username and password will be used for authentication. If authentication is successful, an access_token and refresh_token will be returned
- password (Optional) - Required if grant_type is "password".
- username (Optional) - Required if grant_type is "password".

The JSON response will contain the following parameters:

- access_token - The random 20 character string used to access a user's profile.
- refresh_token - The random 20 character string used to obtain a new access_token. This expires after 24 hrs.
- token_type - currently we only issue bearer tokens.
- expires_in - the number of seconds the token is good for.
- state - Only if this parameter was included in the request.

19.3 Refresh Token

Client app will make a POST request to /id/authenticationService/oauth/refresh. This request will contain the following parame

- client_id (Required) - The client_id your app was issued during when registered.
- client_secret (Required) - The client_secret your app was issued during when registered.
- refresh_token (Required) - The refresh_token you were issued with you access token.

The server will validate the refresh token and if the refresh token is less then 24 hrs old, a new access token will be issued. The current refresh token will be expired and a new one will be issued.

The JSON response will contain the following parameters:

- access_token - The random 20 character string used to access a user's profile.
- refresh_token - The random 20 character string used to obtain a new access_token. This expires after 24 hrs.
- token_type - currently we only issue bearer tokens.
- expires_in - the number of seconds the token is good for.

19.4 API Access

In order to obtain a user's profile information, make a GET request to /id/userService/profile with the access_token as a query parameter.

If the token is still valid, you will receive a JSON response with the following user information:

- firstName
- lastName
- email
- institution

- userId
- username
- projectAdmin
- hasSetPassword

We also support access to any rest services on behalf of the user. Just append “?access_token=your_access_token” to the url in order to access the service.

CHAPTER 20

FIMS v1 migration to FIMS v2

Instructions for updating deployment for nmnh-fims and biscicol-fims.

Run the FimsAlterTableScript in biocode-fims-commons.

To use the FimsAlterTablesScript:

1. copy and paste the "mysql –batch –skip-column-names . . . " to the cmd line.
2. copy the output
3. enter the mysql prompt
4. paste the output.
5. copy and paste the "drop table" and "alter table" lines from the FimsAlterTableScript to the mysql prompt. I recommend doing this in blocks and not just copy and pasting everything at once.

After you complete the db migration, run the biocode.fims.utils.ExpeditionUpdater "main" method in biocode-fims-commons. This will create bcids for all of the expeditions.

CHAPTER 21

REST Service Migration Guide

This lists only the services that were changed. To get more information about the current REST services visit http://biscicol.org/biocode-fims/rest/fims.wadl

/id/authenticationService

- /loginLDAP
  - moved to nmnh-fims
- /entrustChallenge
  - moved to nmnh-fims
- /oauth/access_token
  - moved to "id/authenticationService/oauth/accessToken"
  - additional PostParams:
    - grant_type: optional. If grant_type = "password", then the user will be authenticated with the provided username and password POST params. Thus skipping the need to call "/oauth/authorize" service 1st
    - username: required if grant_type = "password"
    - password: required if grant_type = "password"
- /reset
  - moved to "biocode-fims/rest/users/resetPassword"
- /sendResetToken
  - moved to "biocode-fims/rest/users/{username}/sendResetToken"

/projectService

- /validation/{projectId}
  - removed
- /list

  - moved to “biocode-fims/rest/projects/list”
  - now returns a JSONArray of “project” JSONObjects
    * each JSONObject contains: “projectId”, “projectCode”, “projectTitle”, “validationXml”
- /graphs/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/graphs”
  - now returns a JSONArray of “graph” JSONObjects
    * each JSONObject contains: “expeditionCode”, “expeditionTitle”, “ts”, “identifier”, “bcidId”, “projectId”, “webAddress”, “graph”
- /myGraphs
  - moved to “biocode-fims/rest/projects/myGraphs”
  - “ark” => “identifier”, “expedition_code” => “expeditionCode”, “project_id” => “projectId”, “dataset_id” => “bcidId”, “expedition_title” => “expeditionTitle”
- /myDatasets
  - moved to “biocode-fims/rest/projects/myDatasets”
  - “ark” => “identifier”, “dataset_id” => “bcidId”
- /admin/list
  - moved to “biocode-fims/rest/projects/admin/list”
  - now returns a JSONArray of “project” JSONObjects
    * each JSONObject contains: “projectId”, “projectCode”, “projectTitle”, “validationXml”
- /configAsTable/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/metadata”
- /configEditorAsTable/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/metadataEditor”
- /updateConfig/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/metadata/update”
- /removeUser/{projectId}/{userId}
  - moved to “biocode-fims/rest/projects/{projectId}/admin/removeUser/{userId}”
- /addUser
  - moved to “biocode-fims/rest/projects/{projectId}/admin/addUser”
- /listProjectUsersAsTable/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/users”
- /listUserProjects
  - moved to “biocode-fims/rest/projects/user/list”
  - now returns a JSONArray of “project” JSONObjects
    * each JSONObject contains: “projectId”, “projectCode”, “projectTitle”, “validationXml”
- /saveTemplateConfig
  - moved to “biocode-fims/rest/projects/{projectId}/saveTemplateConfig”

- /getTemplateConfigs/{projectId}
  - moved to "biocode-fims/rest/projects/{projectId}/getTemplateConfigs"
- /getTemplateConfig/{projectId}/{configName}
  - moved to "biocode-fims/rest/projects/{projectId}/getTemplateConfig/{configName}"
- /removeTemplateConfig/{projectId}/{configName}
  - moved to "biocode-fims/rest/projects/{projectId}/removeTemplateConfig/{configName}"

/expeditionService

- /associate
  - moved to "biocode-fims/rest/expeditions/associate"
- /validateExpedition/{projectId}/{expeditionCode}
  - moved to "biocode-fims/rest/expeditions/validate/{projectId}/{expeditionCode}"
- /
  - moved to "biocode-fims/rest/expeditions/"
  - now returns JSONObject of expeditionMetadata
    - "identifier", "public", "expeditionCode", "expeditionTitle", "expeditionId", "projectId"
- /graphMetadata/{graph}
  - moved to "biocode-fims/rest/expeditions/graphMetadata/{graph}"
  - now returns JSONObject of graphMetadata
    - "graph", "projectId", "expeditionOwner", "uploader", "timestamp", "identifier", "resourceType", "finalCopy", "isPublic", "expeditionCode", "expeditionTitle"
- /{projectId}/{expeditionCode}/{resourceAlias}
  - moved to "biocode-fims/rest/expeditions/{projectId}/{expeditionCode}/{resourceAlias}"
  - "ark" => "identifier"
- /deepRoots/{projectId}/{expeditionCode}
  - removed
- /list/{projectId}
  - moved to "biocode-fims/rest/projects/{projectId}/expeditions"
  - now returns JSONArray of "expedition" JSONObjects
  - each "expedition" object contains:
    - "expeditionCode", "expeditionTitle", "public", "expeditionId"
- /resourcesAsTable/{expeditionId}
  - moved to "biocode-fims/rest/expeditions/{expeditionId}/resourcesAsTable"
- /datasetsAsTable/{expeditionId}
  - moved to "biocode-fims/rest/expeditions/{expeditionId}/datasetsAsTable"
- /admin/listExpeditionsAsTable/{projectId}
  - moved to "biocode-fims/rest/projects/{projectId}/admin/expeditions"

- /admin/publicExpeditions
  - moved to “biocode-fims/rest/expeditions/admin/updateStatus”
- /publicExpedition/{projectId}/{expeditionCode}/{publicStatus}
  - moved to “biocode-fims/rest/expeditions/updateStatus/{projectId}/{expeditionCode}/{publicStatus}”

/userService

- /create
  - moved to “biocode-fims/rest/users/admin/create”
- /createFormAsTable
  - moved to “biocode-fims/rest/users/admin/createUserForm”
- /profile/update/{username}
  - moved to “biocode-fims/rest/users/profile/update”
  - now return JSONObject with: “adminAccess”, “returnTo”
- /profile/update
  - moved to “biocode-fims/rest/users/profile/update”
  - now return JSONObject with: “adminAccess”, “returnTo”
- /profile/listEditorAsTable/{username}
  - moved to “biocode-fims/rest/users/admin/profile/listEditorAsTable/{username}”
- /profile/listEditorAsTable
  - moved to “biocode-fims/rest/users/profile/listEditorAsTable”
- /profile/listAsTable
  - moved to “biocode-fims/rest/users/profile/listAsTable”
- /oauth
  - moved to “biocode-fims/rest/users/profile”
  - now return JSONObject with: “firstName”, “lastName”, “email”, “institution”, “hasSetPassword”, “userId”, “username”, “projectAdmin”

/elementService

- /select/{select}
  - moved to “biocode-fims/rest/resourceTypes” or “biocode-fims/rest/resourceTypes/minusDataset”
  - returns a JSONArray of resourceTypes. Each resourceType containing the following: * resourceType, uri, description, string
- /resourceTypes
  - moved to “biocode-fims/rest/resourceTypes”
  - returns a JSONArray of resourceTypes. Each resourceType containing the following:
    - resourceType, uri, description, string
- /creator
  - removed

/groupService

- /
  - moved to “biocode-fims/rest/bcids/”
  - “prefix” => “identifier”
- /metadata/{datasetId}
  - moved to “biocode-fims/rest/bcids/metadata/{bcidId}”
  - returns JSONObject containing “metadataElement” JSONObjects
- /list
  - moved to “biocode-fims/rest/bcids/list”
  - returns JSONArray of “bcid” JSONObjects
  - “identifier”, “bcidId”
- /listUserBCIDsAsTable
  - removed
- /listUserExpeditionsAsTable
  - removed
- /dataGroupEditorAsTable
  - removed
- /dataGroup/update
  - moved to “biocode-fims/rest/bcids/update”

/biocode-fims/rest/authenticationService

- /login
  - no longer uses oAuth to log a user in. Now accepts a username and password to login
- /access_token
  - removed

/mapping

- /filterOptions/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/filterOptions”
  - return JSONArray of JSONObjects * JSONObject contains “column” & “uri” attributes

/query

- /json (GET)
  - moved to “biocode-fims/rest/projects/query/json” (GET)
- / json (POST)
  - moved to “biocode-fims/rest/projects/query/json” (POST)
- /kml (GET)
  - moved to “biocode-fims/rest/projects/query/kml” (GET)
- /kml (POST)
  - moved to “biocode-fims/rest/projects/query/kml” (POST)

- /cspace
  - moved to “biocode-fims/rest/projects/query/cspace”
- /excel (GET)
  - moved to “biocode-fims/rest/projects/query/excel” (GET)
- /excel (POST)
  - moved to “biocode-fims/rest/projects/query/excel” (POST)
- /tab (GET)
  - moved to “biocode-fims/rest/projects/query/tab” (GET)
- /tab (POST)
  - moved to “biocode-fims/rest/projects/query/tab” (POST)

/templates

- /attributes
  - moved to “biocode-fims/rest/projects/{projectId}/attributes “
- /getConfig/{projectId}/{configName}
  - moved to “biocode-fims/rest/projects/{projectId}/getTemplateConfig/{configName}”
- /getConfigs/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/getTemplateConfigs”
- /removeConfig/{projectId}/{configName}
  - moved to “biocode-fims/rest/projects/{projectId}/removeTemplateConfig/{configName}”
- /saveConfig/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/saveTemplateConfig”
- /abstract
  - moved to “biocode-fims/rest/projects/{projectId}/abstract”
  - returns JSONObject with “abstract”
- /createExcel
  - moved to “biocode-fims/rest/projects/createExcel”
- /definition
  - moved to “biocode-fims/rest/projects/{projectId}/getDefinition/{columnName}”

/utils

- /refreshCache/{projectId}
  - removed
- /expeditionCodes/{projectId}
  - moved to “biocode-fims/rest/projects/{projectId}/expeditions”
  - now returns JSONArray of “expedition” JSONObjects
  - each “expedition” object contains:
    - “expeditionCode”, “expeditionTitle”, “public”, “expeditionId”

- /graphs/{projectId}
  - moved to "biocode-fims/rest/projects/{projectId}/graphs"
  - now returns a JSONArray of "graph" JSONObjects
    * each JSONObject contains: "expeditionCode", "expeditionTitle", "ts", "identifier", "bcidId", "projectId", "webAddress", "graph"
- /validateExpedition/{projectId}/{expeditionCode}
  - moved to "biocode-fims/rest/expeditions/validate/{projectId}/{expeditionCode}"
- /getListFields/{listName}
  - moved to "biocode-fims/rest/project/{projectId}/getListFields/{listName}"
  - returns a jsonArray with the acceptable values for the list
- /isNMNHProject/{projectId}
  - removed
- /listProjects
  - moved to "biocode-fims/rest/projects/list"
- /callBCID
  - removed
- /getDatasetDashboard
  - removed
- /updatePublicStatus
  - moved to "biocode-fims/rest/expeditions/updateStatus/{projectId}/{expeditionCode}/{publicStatus}"
- /getLatLongColumns/{projectId}
  - moved to "biocode-fims/rest/projects/{projectId}/getLatLongColumns"

/validate

- /continue_nmnh
  - removed
- /continue_spreadsheet
  - removed