LRI Reboot API Documentation

# General

This document provides specifications for implementing applications that perform search operations on Resources, Standards, and Pathways (also called Learning Maps) in the Learning Resource Index (LRI). This API is not finalized, however, inBloom will endeavor to preserve the published API as much as possible. The most recent version of this document will be found in: <https://github.com/inbloom/lri-reboot/tree/master/doc>

# Environments

The Learning Registry Index sandbox URL is:

<http://maps-sandbox.inbloom.org:8080/lri-reboot/>

# Quick Guide to Searching

Search on resources for Common Core Standard CCSS.Math.Content.8.EE.A.2:

<http://maps-sandbox.inbloom.org:8080/lri-reboot/resources/search?query=resource.standard_t:CCSS.Math.Content.8.EE.A.2>

Search on resources with the tag “Moon”:

<http://maps-sandbox.inbloom.org:8080/lri-reboot/resources/search?query=resource.tag_t:Moon>

Full text search on all indexed fields:

<http://maps-sandbox.inbloom.org:8080/lri-reboot/resources/search?query=resource_solrsummary_t:Smithsonian>

These searches and additional fields are described in detail in the **Resources** section.

# Constructing URLs and payloads

The API is designed as a series of HTTP(S) endpoints that allow you to send and receive JSON payloads. Additionally, you must add an “Accept” header to your request with a value of “application/json”. The basic format of the URL is:

http://{server}:{port}/{path}/{object}/{id}

where:

{server} is the DNS name of the server hosting the LRI software,

{port} is the port number on which the server is running,

{path} is the path to the lri-reboot app,

{object} is the type of object you are requesting, and

{id} is the specific instance of the object.

For example, to request the Standard object with an id of 4 from the sandbox LRI server running on your local machine on port 8080 in the folder lri-reboot, the URL would look like:

<http://maps-sandbox.inbloom.org:8080/lri-reboot/standards/4>

The API is designed in a RESTful style, supporting CRUD operations with the following HTTP methods:

|  |  |
| --- | --- |
| Create | POST |
| Read | GET |
| Update | PUT |
| Delete | DELETE |

Successful POST operations will return a 201 CREATED response including a Location header with a URL for the created object. Future updates will support a full HATEOAS (also known as REST level 3) implementation.

# Note on PUT and DELETE endpoints

Current design indicates PUT and DELETE endpoints that update objects (as opposed to creating relationships between objects) will not be enabled in the production release version of the LRI Reboot. Documentation for PUT and DELETE endpoints should be considered deprecated, and support will likely be removed before release.

# JSONP Support

The LRI supports JSONP on GET endpoints. To request JSONP responses, change the querystring to:

http://{server}:{port}/{path}/{object}/{id}.json?callback={string}

There is an interceptor that will wrap your result in a Javascript function with the name passed in the callback parameter. Note that JSONP is not supported for PUT, POST, and DELETE operations.

Full API Documentation

# AgeRanges

Describes the targeted ages for a Resource object.

## JSON

{  
   "id":1,  
   "name":"0-2",  
   "version":0  
}

## /ageranges

### GET

Retrieves a listing of all AgeRange objects in the system.

### POST

Add a new object.

## /ageranges/{id}

### GET

Retrieves detailed listing for AgeRange object {id}.

# Alignments

Alignment objects manage and describe the relationship between Standards and Resources using an AlignmentType as an attribute of the relationship. You will not usually access this object directly, and it should be created using the PUT operation on the /resources/{id1}/alignmenttypes/{id2}/standards/{id3} endpoint.

## JSON

{

“sample” : “json block”,

“key” : “value”

}

## /alignments

### GET

Retrieves a listing of all Alignments in the system.

## /alignments/{id}

### GET

Retrieves a detailed listing of Alignments {id}.

# AlignmentTypes

Create and maintain a descriptive list of possible alignments between Resources and Standards.

## JSON

{  
   "id":1,  
   "name":"Teaches",  
   "version":1  
}

## /alignmenttypes

### GET

Retrieve a listing of AlignmentTypes.

### POST

Add a new AlignmentType object.

## /alignmenttypes/{id}

### GET

Retrieve a detailed listing of AlignmentType {id}.

# Audiences

Create and maintain a list of possible audiences targeted by Resources.

## JSON

{  
   "id":1,  
   "name":"Administrator",  
   "version":0  
}

## /audiences

### GET

Retrieves a listing of all Audience objects in the system.

### POST

Add a new object.

## /audiences/{id}

### GET

Retrieves a detailed listing of Audience object {id}.

# Interactivity

Describes the interactive nature of a Resource object

## JSON

{  
   "id":1,  
   "name":"Active",  
   "version":0  
}

## /interactivitys

### GET

Retrieves a listing of all Interactivity objects.

### POST

Add a new object.

## /interactivitys/{id}

### GET

Retrieves a detailed listing of Interactivity object {id}.

# Languages

Lists languages and their abbreviated codes.

## JSON

{  
   "code":"en",  
   "id":1,  
   "name":"English",  
   "version":0  
}

## /langs

### GET

Retrieve a listing of all languages in the system.

### POST

Add a new object.

## /langs/{id}

### GET

Retrieve a detailed listing of Language object {id}.

# LearningMaps

Stores a URL-encoded JSON blob describing a Learning Map. Used by the inBloom Learning Maps Authoring Tool, documented separately.

## JSON

{  
   "id":1,  
   "name":"My learning Map",  
   "payload":"%7B%0A%20%20%20%22key1%22%3A%22value1%22%2C%0A%20%20%20%22key2%22%3A%22value2%22%0A%7D",  
   "standard":{  
      "externalId":"Top Level Standard",  
      "heading":"Heading",  
      "id":1,  
      "name":"Top Level Standard",  
      "parent":null,  
      "standard\_text":"This is the text of the standard",  
      "subheading":"Subheading",  
      "url":"http://www.standards.org/top",  
      "version":0  
   },  
   "version":0  
}

## /learningmaps

### GET

Retrieves all LearningMap objects in the system. This query may return large amounts of data.

### POST

Add a new object.

## /learningmaps/{id}

### GET

Retrieves a single LearningMap object identified by {id}.

# LearningResources

Describes the format types possible for Resource objects.

## JSON

{  
   "id":1,  
   "name":"Audio CD",  
   "version":0  
}

## /learningresources

### GET

Retrieve a listing of all possible LearningResource types.

### POST

Add a new LearningResource type.

## /learningresources/{id}

### GET

Retrieve a detail listing of LearningResource {id}.

# Parties

Describes an individual or organization that interacts with a Resource in some way, as a creator, rights holder, user, etc.

## JSON

{  
   "id":1,  
   "name":"Verlin Henderson",  
   "version":0  
}

## /partys

### GET

Retrieve all Party objects in the system.

### POST

Add a new Party object.

## /partys/{id}

### GET

Retrieve a detailed listing for Party objects {id}.

# PartyTypes

Lists possible roles played by Party objects to describe their association with a Resource.

## JSON

{  
   "id":1,  
   "name":"Author",  
   "version":0  
}

## /partytypes

### GET

Retrieves all PartyTypes.

### POST

Add a new object.

## /partytypes/{id}

### GET

Retrieves PartyType identified by {id}.

# Resources

## JSON

{

"URL" : "http://www.sample.com",

"copyrightYear" : "2013",

"description" : "An example resource",

"externalGUID" : "GUID",

"id" : 1,

"interactivity" : {

"id" : 1,

"name" : "Active",

"version" : 0

},

"isBasedOnURL" : "http://www.sample2.org",

"lang" : {

"code" : "en",

"id" : 1,

"name" : "English",

"version" : 0

},

"learningResource" : {

"id" : 1,

"name" : "Audio CD",

"version" : 0

},

"name" : "Sample Resource",

"sourceText" : "Sample Source Text",

"timeRequired" : "30 minutes",

"useRightsURL" : "http://www.sample1.org",

"version" : 0

}

## /resources

### GET

### POST

Add a new Resource object.

## /resources/{id}

### GET

Retrieve detailed information for a single Resource object identified by {id}.

## /resources/search?query={query}

### GET

Access to a search API powered by Apache Solr. Based on the query string parameter, returns a collection of resources that match the query.

You can construct a query for the following items, which correspond to properties of the Resource object (detailed in the JSON above):

* id
* resource.name\_s
* resource.externalguid\_s
* resource.url\_s
* resource.description\_s
* resource.copyrightyear\_s
* resource.userightsurl\_s
* resource.isbasedonurl\_s
* resource.timerequired\_s
* resource.lang\_t
* resource.audience\_t
* resource.agerange\_t
* resource.use\_t
* resource.interactivity\_t
* resource.tag\_t
* resource.party\_t
* resource.standard\_t
* resource.sourcetext\_s
* resource.id\_l
* resource\_solrsummary\_t

Items which end in \_s are String types and must be an exact match (including wildcards). Items which end in \_l are Long types. Items which end in \_t are Text types and can be searched with a case insensitive search (including wildcards).

Searches for the resource.party\_t item must be prefaced with the type of party you are searching for, followed by a “:” character. (Ex: Publisher, Author, Editor)

Searches for standards must be prefaced with an alignment type, followed by a “:” character. (Ex: Teaches, Assesses)

For more information about constructing Solr queries, refer to the Solr Wiki page on query syntax: <http://wiki.apache.org/solr/SolrQuerySyntax>.

If you use JSONP with this endpoint, construct your request as:

/resources/search.json?callback={callback}&query={query}

# Standards

Learning standards are organized into a tree of Standards objects, which describe the different levels of a learning standard.

## JSON

{  
   "externalId":"Top Level Standard",  
   "heading":"Heading",  
   "id":1,  
   "name":"Top Level Standard",  
   "parent":null,  
   "standard\_text":"This is the text of the standard",  
   "subheading":"Subheading",  
   "url":"http://www.standards.org/top",  
   "version":0  
}

## /standards

### GET

Get all standards in the system with no parent standard, meaning that this endpoint will return all top-level standards in the LRI. This behavior is different from other top-level endpoints, which return all associated objects.

### POST

Add a new top-level standard.

## /standards/{id}

### GET

Get details on a specific standard {id}.

### POST

Create a new standard with parent standard {id}.

### PUT

Update an existing standard {id}.

### DELETE

Delete the standard {id}.

## /standards/{id}/pathways

### GET

List pathways that contain nodes referencing this standard

## /standards/search?query={query}

### GET

Access to a search API powered by Apache Solr. Based on the query string parameter, returns a collection of standards that match the query.

You can construct a query for the following items, which correspond to properties of the Standard object (detailed in the JSON above):

* standard.name\_s
* standard.url\_s
* standard.externalid\_s
* standard.parent\_t
* standard.heading\_s
* standard.subheading\_s
* standard.standard\_text\_s
* standard.id\_l
* standard\_solrsummary\_t

Items which end in \_s are String types and must be an exact match (including wildcards). Items which end in \_l are Long types. Items which end in \_t are Text types and can be searched with a case insensitive search (including wildcards). For more information about constructing Solr queries, refer to the Solr Wiki page on query syntax: <http://wiki.apache.org/solr/SolrQuerySyntax>.

If you use JSONP with this endpoint, construct your request as:

/standards/search.json?callback={callback}&query={query}

# Tags

## JSON

{  
   "id":1,  
   "name":"Sample Tag",  
   "version":0  
}

## /tags

### GET

Retrieves a listing of all Tag objects.

### POST

Add a new Tag object.

## /tags/{id}

### GET

Retrieves a detailed listing of Tag object {id}.

# Use

Describes the uses for an associated Resource.

## JSON

{  
   "id":1,  
   "name":"Activity",  
   "version":0  
}

## /uses

### GET

Retrieve a listing of all possible Uses.

### POST

Add a new Use object.

## /uses/{id}

### GET

Retrieve a detailed listing for Use {id}.