[Redacted] API Reference

Version 2.3

REDACTED

REDACTED

Legal Notices

Contents

[Redacted] API **1**
[Redacted] system overview 1
Participation 2
API authentication **3**
Authentication errors 4
Third-party authentication 4
HTTP header elements **5**
Request properties 6
Response structure **7**
Response content 7
Local server, agency, and location **10**
System configuration **12**
Delete a Single Location 12
Get Central Item Types 13
Get Central Patron Types 14
Get Local Servers 16
Retrieve Locations List 18
Upload Locations List 19
Upload/Update a Single Location 20
Glossary **22**

[This page blank for double-sided printing.]

[Redacted] API

This reference manual describes methods used to facilitate communication between different [redacted] system components. [Redacted] enables participating libraries to offer materials to other libraries in a consortia borrowing system. Patrons of participating libraries can directly request and borrow materials from other participating libraries through the union catalog. To help facilitate this, [redacted] provides a set of application programming interfaces (APIs) that:

- circulate material between participating sites
- update information related to cataloged material
- manage records stored in the union catalog and participant local catalogs

In addition to API functionality, this reference also discusses security, authentication, and authorization.

[Redacted] system overview

The following diagram depicts how communication is passed between different [redacted] system components in an [redacted] consortium. As shown below, multiple consortia are accessible from the same [redacted] service. Both the [redacted] and [redacted] ILSs are clustered near the consortia they join.

REDACTED

Libraries participating in an [redacted] consortium contribute records to the central union catalog. When local records are updated, those in the union catalog are also updated. When patrons request materials from the union catalog, appropriate transactions are recorded in both borrowing and lending (owning) local libraries, and the record status is updated in both the local catalog and the union catalog. The [redacted] central server performs patron validation at the local library. All transactions are accomplished automatically, in real time.

The [redacted] API allows libraries using any [redacted] integrated library software system to participate in an [redacted] resource-sharing consortium. For more information, see Participation, below.

See Glossary for definitions of [redacted] terms.

Participation

If you use a [redacted] integrated library software system and are interested in participating in an [redacted] resource-sharing consortium, contact [redacted].

You will need to provide authentication information to try the [redacted] API. See API authentication on page 3.

API authentication

[Redacted] API requests are authenticated with an oauth2 token. An API key and secret are used to request access tokens from the [redacted] platform.

The oauth2 endpoint uses the POST method to obtain access tokens. The client credential is provided in the HTTP authorization header with an HTTP basic authentication type. Both "grant_type" and "scope" are required parameters. The grant type is “client_credential” while “[redacted]” is the scope for third-party users.

The following cURL sample requests an access token from our sandbox environment.

```
curl -X POST https://rssandbox-api.iii.com/auth/v1/oauth2/token -H
'Authorization: Basic YjU0ZTU2YzgtMGNlNi00MzhjLTk3NzktYzcyMWIxY2RjMz
ZhOjFjYjQ4YjNmLTE1OTAtNDc0YS1iMDhhLWZjNDRlMjczMTlmOQ=='
   -H 'Content- Type: application/x-www-form-urlencoded'
   -d      'grant_type=client_credentials&scope=innreach_tp'
```

Sample response:

```
{
   "access_token": "06b42df732f628ae2c3764d86073cd76",
   "token_type": "Bearer",
   "expires_in": 599
}
```

The provided key and secret should be concatenated using a colon separator, and the resultant string base64 encoded.

The access token is valid for API requests until it expires in 600 seconds. An expired token will result in a “401 Unauthorized” HTTP status code in the response.

Notes: The oauth2 endpoint to get an access token on the production environment is issued only upon request.

Authentication errors

When requesting access tokens, any missing or incorrect required property will result in a “400 Bad Request” in the response. A “401 Unauthorized” error is issued if client credentials are provided incorrectly.

```
400 Bad Request
{
   "error": "invalid_request",
   "error_description": "The scope parameter is required."
}

401 Unauthorized
{
   "error": "invalid_token",
   "error_description": "Token authentication failed"
}
```

Third-party authentication

To secure an API, third-party developers must take the same approach as described in API Authentication above. An equivalent oauth2 endpoint needs to be provided to produce a bearer token from the [redacted] platform.

To join an [redacted] central consortium, the following information must be provided during setup.

- The oauth2 endpoint to get an access token for requesting the APIs implemented by the third party.
- The [redacted] credentials used to generate the access token.
- [redacted] parameters required to specify when calling the oauth2 endpoint. For example: `grant_type=client_credentials&scope=[redacted]_tp`

The following is an example call that would originate from [redacted] to the third party.

```
curl -X POST http://rd-mock.[redacted].com/[redacted]/v2/oauth2/
token -H 'Accept: application/json'
   -H 'Authorization: Basic
   ZmQ1OTdlXmItMjNkLS00ZWZiLAE0ZjUtN2Y2NJFhNDc2Njk2OjVhMjcxMjdhLUk0YmUtNG
   JhYi05MDMzLWI1MmFjMzkxOTQ3Yg=='
   -H 'Content-Type: application/x-www-form-urlencoded'
   -d
   'grant_type=client_credentials&scope=[redacted]_tp'
```

HTTP header elements

The following HTTP headers are necessary for [redacted] API methods.

Property	Type	Required	Description
Accept	String	Yes	Acceptable value: application/json
Authorization	String	Yes	Authorization in form <token_type> <access_token> as returned by the authorization request.
Content-Type	String	Yes	Acceptable value: application/json
X-From-Code	String	Yes	Central/local code of sending system (five-character code). The library needs only its own local code and the code for any central systems in which it participates.
X-Request-Creation-Time	Integer	Yes	Epoch UNIX time stamp, for example: 1544466568
X-To-Code	String	Yes	Central/local code of destination system (five-character code). The library needs only its own local code and the code for any central system in which it participates.

Request properties

Request properties are sent as an HTTP request body in JSON format, which is the only format currently supported in [redacted].

Example: ContributeBatch Item

```
{
   "itemInfo":[
      {
         "agencyCode":"plag2",
         "callNumber":"940.53 Ien Ien",
         "copyNumber":null,
         "dueDateTime":null,
         "holdCount":0,
         "itemCircStatus":"Available",
         "itemId":"2475459",
         "itemNote":null,
         "centralItemType":14,
         "locationKey":"test1",
         "marc856PublicNote":null,
         "marc856URI":null,
         "suppress":"n",
         "volumeDesignation":null
      }
   ]
}
```

Example: Item Shipped

```
{
   "itemBarcode": "ibare34"
}
```

Response structure

[redacted] responses use HTTP status code and content body to inform API users of the request processing status.

HTTP Status Code	Description	[redacted] Meaning
200	OK	All request properties are valid and successfully serviced in the central server. However, the final request status can still be "failed" due to other reasons.
400	Bad Request	The request either does not contain all the required properties or has invalid value(s) in one or more properties.
401	Unauthorized	The request does not have the appropriate authentication token in the HTTP Authorization header.
405	Method Not Allowed	The request is not submitted with the correct HTTP method. For example, using GET instead of POST as required by the specific API.
500	Internal Server Error	The central encountered problems that are not one of the above while servicing the request.
501	Not Implemented	The server does not support the functionality to fulfill the request.
503	Service Unavailable	The server is unavailable.

Response content

The response content is a JSON-formatted string.

Contribution requests are processed synchronously. The response status reflects the status of the request.

Circulation requests are validated and submitted to the central queue for processing. The response status is *not* the final status of the request. An "`ok`" status means the request processing queue accepted the request.

Property	Description
status	Either "ok" or "failed" for a string type.
reason	A short description about the final request status.
errors	The list of problems encountered in servicing the request.

Failed response examples

```
{
   "status":"failed",
   "reason":"Authorization token does not belong to 'tploc'",
   "errors":[]
}
```

```
{
   "status":"failed",
   "reason":"Unknown centralCode and trackingId combination",
   "errors":[
      {
         "type":"FieldError",
         "reason":"Invalid record key",
         "name":"centralCode",
         "rejectedValue":"pcent"},
      {
         "type":"FieldError",
         "reason":"Invalid record key",
         "name":"trackingId",
         "rejectedValue":"83748742"
      }
   ]
}
```

```
{
   "status":"failed",
   "reason":"Invalid request",
   "errors"[
{
   "type":"FieldError",
   "name":"patronAgencyCode",
   "rejectedValue":"plag1234",
   "reason":"Invalid Field Value"
},
{
   "type":"FieldError",
   "name":"patronName",
   "rejectedValue":null,
   "reason":"Field missing"
},
{
   "type":"HeaderError",
   "name":"X-From-Code",
   "rejectedValue":"ploc8",
   "reason":"Invalid Header Value"
},
{
   "type":"HeaderError",
   "name":"X-To-Code",
   "rejectedValue":"d2ir",
   "reason":"Invalid Header Value"
}
```

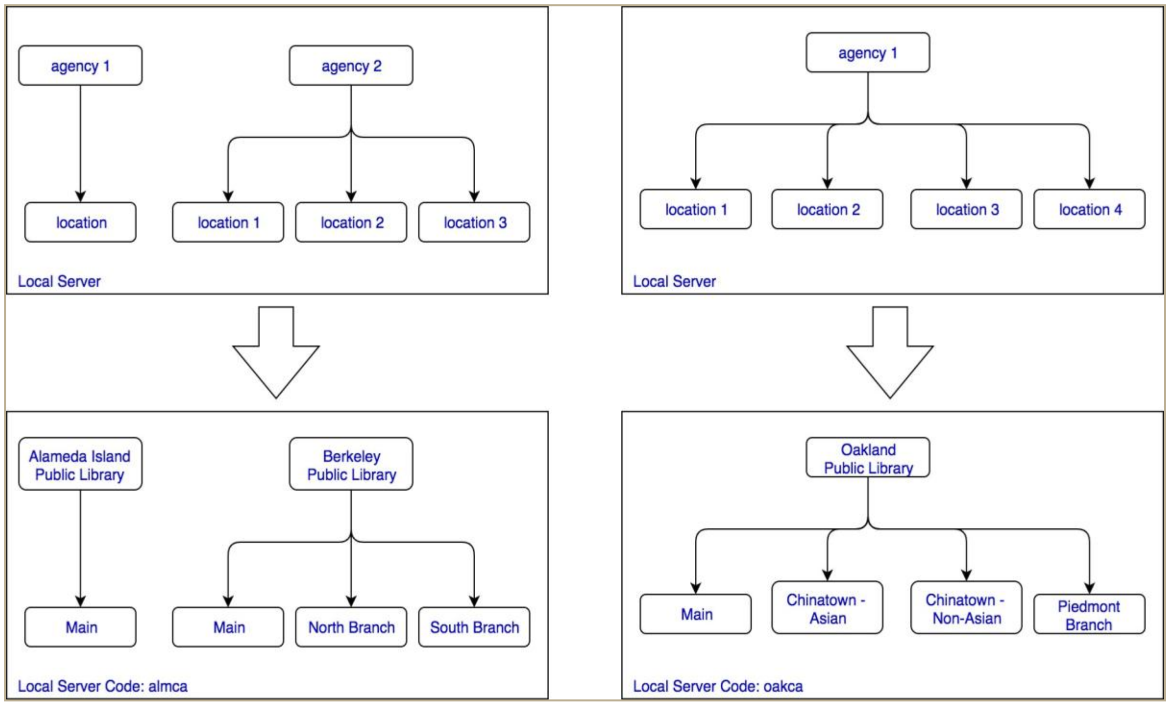
Local server, agency, and location

This section discusses the relationship between the local server, agency, and location and how these work within an ILS and associated consortium.

These three elements within a consortium are defined as follows:

- **Local server:** The computer assigned to an ILS that manages one or more libraries in the consortium.
- **Agency:** A mapping that contains a library or multiple libraries managed by a [redacted] ILS connected to an [redacted] Central system.
- **Location:** An ILS-defined name that represents one or more library branches or shelving locations.

As shown in the diagram below, an agency is mapped to a library when multiple libraries are managed by a [redacted] ILS in an [redacted] Central system. When a library is mapped to an agency, its location is mapped to library branches. Each branch may contain shelving locations.



A local server code is assigned to an ILS that manages one or more libraries. Each of these libraries may contain multiple branches. An agency maps to a library to differentiate it from other libraries within the [redacted] Central system. Foreach library mapped to an agency, its location is assigned to library branches and their associated shelving locations.

The diagram above shows the local server code assigned to an ILS. The ILS can manage one or more libraries with each library possibly containing multiple branches.

The agency, associated with a library, is assigned an agency code that must be unique within the consortium. Unlike the local server code or agency code, locations are defined by the ILS and represent library branches or shelving locations.

A location contains a location key and description field. If the location key is within the same consortium, it does not need to be unique. A locationKey within the same local server must be unique and cannot share the same name as another locationKey on the server. In all cases, the agency code needs to be unique, since the local server can participate in multiple consortia.

Before a record can be contributed to the Central, the ILS needs to define each location and upload them to the Central, since the location key is required for the Contribute Batch Items API.

System configuration

Local sites can configure locations. A list of central item types, central patron types, and local servers can also be queried.

The APIs in this section include:

- Delete a Single Location on page 12
- Get Central Item Types on page 13
- Get Central Patron Types on page 14
- Get Local Servers on page 16
- Retrieve Locations List on page 18
- Upload Locations List on page 19
- Upload/Update a Single Location on page 20

Delete a Single Location

Deletes a single location matching <locationKey>.

DELETE /[redacted]/v2/location/<locationKey>

Notes: A URL can be submitted with a location key that matches a previously deleted location. This reinstates the location with the new description and original key.

URL parameters

Property	Description
locationKey	Unique code that represents the location being deleted. Lowercase alphanumeric, maximum 5 characters.

Request parameters

None

Sample cURL call

```
curl -X DELETE http://{host}:{port}/[redacted]/v2/location/{locationKey} -
H 'Accept: application/json'
   -H 'Authorization: Bearer f071cb6c9220434c0e349cd9d69892e9'
   -H 'X-From-Code: {localcode}' -H 'X-To-Code: {centralcode}'
```

Get Central Item Types

An [redacted] "item type" combines material type and loan rule/period to determine who can borrow an item and forhow long. Central item types are required for item contribution.

This method requests a list of central server item type values with descriptions, which are required by a local library to set up a mapping table between local item types and those accepted by central.

GET /[redacted]/v2/contribution/itemtypes

URL parameters

None

Response content

Property	Type	Required
status	String	Yes
reason	String	Yes
errors	[ ]	Yes
ItemTypeList	[ ]	Conditional

ItemTypeList

Property	Type	Required
centralItemType	Integer	Yes
description	String	Yes

Sample cURL call

```
curl -X GET http://{host}:{port}/redacted/v2/contribution/itemtypes -H
'Accept: application/json'
   -H 'Authorization: Bearer 741619d4e1b2156b98e084a08b5a00d9'
   -H 'X-From-Code: {localcode}'
   -H 'X-To-Code: {centralcode}'
```

Sample Response:

```
{
   "status":"ok",
   "reason":"success",
   "errors":[],
   "itemTypeList":[
      {
         "centralItemType":0,
         "description":"[redacted] zero itype"
      },
     ...
      {
         "centralItemType":255,
         "description":"Poof and piffle 12/7 5:25"
      }
   ]
}
```

Get Central Patron Types

Requests a list of central server patron type values with descriptions, which are required by a local library to set up a mapping table between their local patron types and those acceptable by central.

```
GET /[redacted]/v2/circ/patrontypes
```

Note: An [redacted] *patron type* is required to determine who can borrow an item and for how long.

URL parameters

None

Request parameters

None

Response content

Property	Type	Required
status	String	Yes
reason	String	Yes
errors	[ ]	Yes
patronTypeList	[ ]	Conditional

patronTypeList

Property	Type	Required
centralPatronType	Integer	Yes
description	String	Yes

Sample cURL call

```
curl -X GET http://{host}:{port}/redacted/v2/circ/patrontypes -H
'Accept: application/json'
   -H 'Authorization: Bearer 2af564e268eae1375c44e8c25fc77348'
   -H 'X-From-Code: {localcode}'
   -H 'X-To-Code: {centralcode}'
```

Sample Response:

```
{
   "status": "ok",
   "reason": "success",
   "errors": [],
   "patronTypeList": [
      {
      "centralPatronType": 15,
      "description": "[redacted] ptype 15"
      },
      …
      {
      "centralPatronType": 210,
      "description": "Central two ten"
      }
   ]
}
```

Get Local Servers

Requests a list of all local server agency codes with descriptions.

```
GET /[redacted]/v2/contribution/localservers
```

URL parameters

None

Request parameters

None

Response content

Property	Type	Required
status	String	Yes
reason	String	Yes
errors	[ ]	Yes
localServerList	[ ]	Optional

localServerList

Property	Type	Required	Description
localCode	String	Yes	5 character code
description	String	Yes	128 character maximum
agencyList	[ ]	Optional	

agencyList

Property	Type	Required	Description
agencyCode	String	Yes	5 character code
description	String	Yes	128 character maximum

Sample cURL call

```
curl -X GET http://{host}:{port}/[redacted]/v2/localservers
   -H 'Accept:application/json'
   -H 'Authorization:Bearer2af564e268eae1375c44e8c25fc77348'
   -H 'X-From-Code:{localcode}'
   -H 'X-To-Code:{centralcode}'
```

Sample Response:

```
{
   "status": "ok",
   "reason": "success", "errors": [],
   "localServerList": [
      {
         "localCode": "9irv1",
         "description": "irv system",
         "agencyList": [
            {
               "agencyCode": "9aaaa",
               "description": "Appalachian"
            },
            {
               "agencyCode": "9bbbb",
               "description": "Irv Big Bend"
            }, ...
         ],
      },
      {
         "localCode": "9loc1",
         "description": "IRBOX 1, irdvloc1",
         "agencyList": [
            {
               "agencyCode": "9bada",
               "description": "Bad Axe Public Library"
            },
            {
               "agencyCode": "9bayl",
               "description": "Bayliss Public Library"
            },
            ...
         ],
      },
      ...
   ]
 }
```

Retrieve Locations List

Requests a list of submitted location codes with descriptions.

GET /[redacted]/v2/contribution/locations

URL parameters

None

Response content

Property	Type	Required
status	String	Yes
reason	String	Yes
errors	[ ]	Yes
locationList	[ ]	Yes

locationList

Property	Type	Required	Description
locationKey	String	Yes	Unique code that represents the location. Lowercase alphanumeric, maximum 5 characters.
description	String	Yes	255 char max

Sample cURL request

```
curl -X GET http://{host}:{port}/[redacted]/v2/contribution/locations -H
'Accept: application/json'
   -H 'Authorization: Bearer 39d0854c74df7ddf9a54ad56fa9af15c'
   -H 'X-From-Code: jclcl'
   -H 'X-To-Code: pcent'
```

Sample response:

```
{
   "status":"ok",
   "reason":"success",
   "locationList":[
      {
         "locationKey":"main",
         "description":"Main Library"
      },
   ...
      {
         "locationKey":"wcdpl",
         "description":"Wood County District Public Library"
      }
   ],
   "errors":[]
}
```

Upload Locations List

Submits a list of location keys with descriptions. Replaces existing locations with a new set.

POST /[redacted]/v2/contribution/locations

URL parameters

None

Request parameters

Property	Type	Required
locationList	[ ]	Yes

locationList

Property	Type	Required	Description
locationKey	String	Yes	Unique code that represents the location. Lowercase alphanumeric, maximum 5 characters.
description	String	Yes	255 char max

Sample cURL call

```
curl -X POST http://{host}:{port}/[redacted]/v2/contribution/locations - H
'Accept: application/json'
   -H 'Authorization: Bearer b50cb855e0a24c69f6fe-b14dd6282f95'
   -H 'Content-Type: application/json' -H 'X-From-Code: {localcode}'
   -H 'X-To-Code: {centralcode}'
   -d '{ "locationList": [
         {
         "locationKey":"main",
         "description":"JCLCL Main Library"
      },
      {
         "locationKey":"dnvl",
         "description":"JCLCL Danville Branch" }
   ]
   }
```

Upload/Update a Single Location

Adds a single new location or updates the description of an existing location with a matching <locationKey>.

POST	/[redacted]/v2/location/<locationKey>
PUT	/[redacted]/v2/location/<locationKey>

URL parameters

Property	Description
locationKey	Unique code that represents the location being uploaded or updated. Lowercase alphanumeric, maximum 5 characters.

Request parameters

Property	Type	Required	Description
description	String	Yes	255 char max

Sample cURL calls

POST creates a new location or updates and existing location.

```
curl -X POST http://{host}:{port}/[redacted]/v2/location/{locationKey} -H
'Accept: application/json'
   -H 'Authorization: Bearer 7c7b93073f829c24a6a2d628005cfe1d'
   -H 'Content-Type: application/json'
   -H 'X-From-Code: {localcode}'
   -H 'X-To-Code: {centralcode}'
   -d '{
      "description": "JCLCL Danville Branch MOD..."
      }'
```

PUT updates an existing location.

```
curl -X PUT http://{host}:{port}/[redacted]/v2/location/{locationKey} -H
'Accept: application/json'
   -H 'Authorization: Bearer 7c7b93073f829c24a6a2d628005cfe1d'
   -H 'Content-Type: application/json'
   -H 'X-From-Code: {localcode}'
   -H 'X-To-Code: {centralcode}'
   -d '{
      "description": "JCLCL Danville Branch MOD ..."
      }'
```

Failed status response

The following error returns if an incorrect or nonexistent locationKey is used.

```
{
   "status": "failed",
   "reason": "Record not found",
   "errors": []
}
```

[*API method review samples end here.*]

Glossary

Agency

Collections of bibs, serials holdings, items, and patrons are defined as belonging to an organization (agency) on a Local Server, which may host multiple agencies. Each agency is identified as a separate organization on the [redacted] Catalog. Because it is assumed that all agencies on a Local Server share the same namespace, record IDs must be unique within the Local Server. When patrons request items from the [redacted] Catalog, they must identify the agency to which they belong.

Agency Code

Five-character code that represents an agency. The code must be unique among all agencies using [redacted], regardless of the resource sharing consortium in which the agency participates. [redacted] helps [redacted] customers determine that the agency code is unique.

[redacted] Catalog

A combined database of the holdings from all the Local Servers that contribute to the [redacted] Central Server. The [redacted] Catalog comprises master bibliographic records and detailed holdings from the sites that own copies of those records. The [redacted] Central Server receives records and updates from Local Servers in real time.

[redacted] Central Server

Houses and provides Discovery for the [redacted] Catalog. Also serves as the hub through which circulation requests and messages are received and sent to owning and borrowing sites. Uses [redacted] software to perform a variety of tasks, including building and maintaining the [redacted] Catalog; building and maintaining [redacted] statistics; load balancing; and determination of lender.

[redacted] Request

Request for an item on the [redacted] Catalog.

[redacted] System

A group of Local Servers and the [redacted] Central Server to which they all contribute. Local Servers that contribute to more than one [redacted] Central Server belong to more than one [redacted] System.

Local Server

Stores records that are contributed to an [redacted] Central Server. A Local Server is often the integrated library system for the libraries it represents.

Local Server Code

A five-character code that represents the Local Server. The code must be unique among all Local Servers using [redacted], regardless of the resource sharing consortium in which the Local Server participates. [redacted] helps [redacted] customers determine that the Local Server code is unique.

Non-[redacted] Libraries

Libraries that do not use [redacted] integrated library systems for their OPAC or circulation modules.

Owning/Lending Site

The site to which the requested [redacted] item belongs.

Patron/Borrowing Site

The site to which the requesting patron belongs.

Site

Generic term that represents an agency on a Local Server.