Queens Library API Requirements Document For e-Content Partners

Version	Date	Author	Description
1	08/03/2015	Surinder Pal Singh	Draft
1.1	08/07/2015	Surinder Pal Singh	Revised by Team
1.2	08/19//2015	Surinder Pal Singh	Revised by Team
1.3	08/20/2015	Roy Pellicano	Additional revisions to the summary text
1.4	02/08/2016	Ankaj Patidar	Revised based on feedback from partners and interested parties

Application Programming Interfaces (API) Standards for Libraries (Digital Materials)

Queens Library API Requirements Document For e-Content Partners

Overview

Queens Library is working towards enabling customers to use library content and services using our e-Resources. For that we are engaging in extensive development to provide them with better user experience and applications (Mobile & Web). These applications integrate with the widest variety of internal systems. For complete integration and a unified and simple user experience we need our e-Content providers to provide us with a flexible API framework, which is detailed in the document below. APIs are becoming increasingly important as content delivery is becoming more dynamic and many new devices generate and consume data via an ever-expanding list of Web services.

Operations

API requests are standard HTTPS requests against vendor resources. In general supported operation are GET, POST, PUT, DELETE, HEAD

- GET queries for a list of content or the details of a specific item.
- POST creates a new content and will provide confirmation in the response body.
- PUT updates an existing content/item with the specified parameters.
- DELETE removes or terminates or deactivates a content record
  (This is on vendor discretion to deactivate or permanently delete).
- HEAD provides response headers, including a count of matching resources.

We provide REST API's and encourage/expect our vendors to provide REST API's for integration. API developers should also make sure that everything is transmitted through the Secure Socket Layer (SSL) - encrypted and transmitted by HTTPS - so that information like usernames and passwords are not captured in-process and then used to gain access to users' accounts or worse, the host organization's account or in other words I can say combat the possibility of "man in the middle attack".

API Response (JSON)

We expect all API calls to provide return response as JSON object. In the 'old' days, web services used XML as their primary data format for transmitting back data, but since JSON appeared (*The JSON format is specified in RFC 4637 by Douglas Crockford*), it has been the preferred format because it is much more lightweight.

What is JSON?

JavaScript Object Notation (JSON) is a lightweight data-interchange format inspired by the object literals of JavaScript.

JSON values can consist of:

objects (collections of name-value pairs) arrays (ordered lists of values) strings (in double quotes) numbers true, false, or null, JSON is language independent.

JSON with PHP?

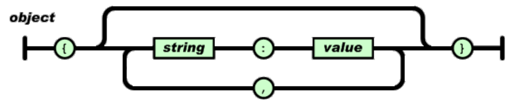
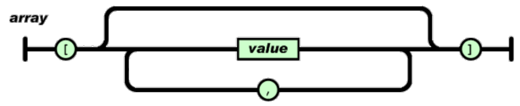
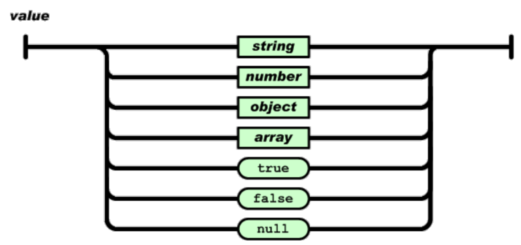
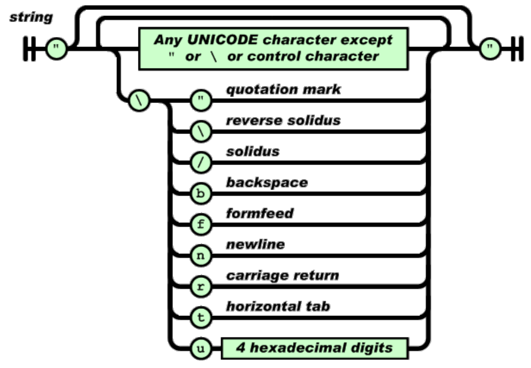
After PHP Version 5.2.0, JSON extension is decodes and encodes functionalities as default.
Json_encode - returns the JSON representation of values Json_decode - Decodes the JSON String Json_last_error - Returns the last error occured.

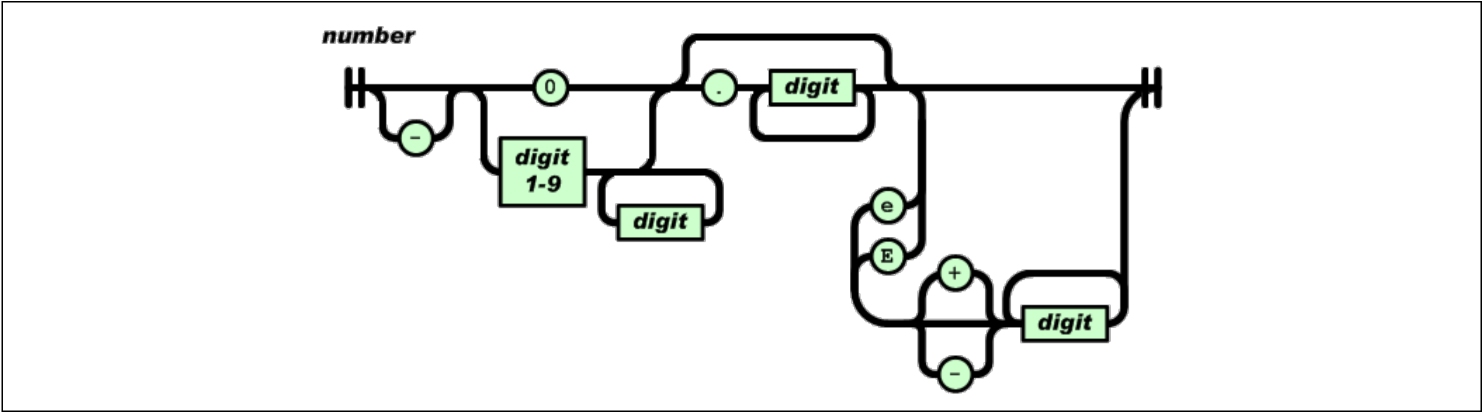
JSON Syntax and Rules?

JSON syntax is derived from JavaScript object notation syntax:
Data is in name/value pairs Data is separated by commas Curly braces hold objects Square brackets hold arrays

JSON is built on two structures:

A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

	
 value string number object array true false null	 string " Any UNICODE character except " or \ or control character \ " quotation mark \ reverse solidus / solidus b backspace f formfeed n newline r carriage return t horizontal tab u 4 hexadecimal digits "



Images used in the illustration are from http://json.org/

Example of JSON Data

```
{
    "firstName": "Surinder",
    "lastName": "Singh",
    "address": {
       "streetAddress": "121 20th Street ",
       "city": "New York",
       "state": "NY",
       "postalCode": 11418
    },
    "phoneNumbers": [
       "555 555-5555",
       "666 555-6666"
    ]
}
```

Conventions

- **Client** - Client application.
- **Status** - HTTPS status code of response.
- All the possible responses are listed under 'Responses' for each method. Only one of them is issued per request server.
- The types of values accepted for a request parameter are shown in the values column.

Status Codes

All status codes are standard HTTPS status codes. The below ones are used in this API.
2XX - Success of some kind
4XX - Error occurred in client's part
5XX - Error occurred in server's part

Status Code	Description
200	OK
201	Created
202	Accepted (Request accepted, and queued for execution)
400	Bad request
401	Authentication failure
403	Forbidden
404	Resource not found
405	Method Not Allowed
409	Conflict
412	Precondition Failed
413	Request Entity Too Large
500	Internal Server Error
501	Not Implemented
502	Connection Timeout
503	Service Unavailable

Working with Business Objects

What kind of API's we expect from e-Content vendors to do our day-to-day activities.

API Entity	Create	Read	Update	Delete
Institutional Authentication	✗	✓	✗	✗
Login/Authenticate	✗	✓	✗	✗
Patron Account Information	✗	✓	✗	✗
Replace Patron Barcode	✗	✗	✓	✗
Checkout Item	✓	✗	✗	✗
Streaming Audio / Video / Online Reading	✗	✓	✗	✗
Retrieve Check-In Information	✗	✓	✗	✗
Data Store Retrieval	✗	✓	✗	✗
Remove Unlicensed Items Data Store	✗	✓	✗	✗
Item Availability	✗	✓	✗	✗
Hold/Request Item	✓	✗	✗	✗
Cancel Held/Requested Item	✗	✗	✗	✓
Search	✗	✓	✗	✗
Ranked Featured Items	✗	✓	✗	✗
Register Patron with Vendor	✓	✗	✗	✗
Synchronize/Change Password across Vendors	✗	✗	✓	✗

Institutional Authentication

The OAuth 2.0 Authorization Framework is widely used to grant access to protected resources. It defines several Authentication flows that all follow the same principle: in exchange of an authorization grant, the client receives a token to access protected resources. For us Vendor uses oAuth Protocol or QL's proposed simple authentication, in both cases we will expect an authorization token / key which will be used in all further API calls for the same Patron in the same session. This API Authenticates Library/Institution with the vendor. This is to be used when a SaaS vendor will need to authenticate as Queens Library with other Library Vendors, for example: if Vendor A needs to login to Vendor B's SaaS service as Queens Library.

Process URL

Type	Values
GET	`https://<baseurl>/login/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String

Expected Response

Status	Response
200	`{ “auth_key”: encoded String or <null> “status” : Boolean true or false }`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`
401	`{"error":"Invalid Library Id."}`

Login / Authentication

The OAuth 2.0 Authorization Framework is widely used to grant access to protected resources. It defines several Authentication flows that all follow the same principle: in exchange of an authorization grant, the client receives a token to access protected resources. For us Vendor uses oAuth Protocol or QL's proposed simple authentication, in both cases we will expect an authorization token / key which will be used in all further API calls for the same Patron in the same session. This API Authenticates Library/Institution with the vendor. This API logs in and/or authenticates a Library Patron with the Vendor. This authenticates the user with the Vendor and obtains the auth_key that Queens Library will use in all patron related API calls.

Process URL

Type	Values
GET	`https://<baseurl>/login/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	patron_id	String
POST	password	String

or

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	auth_token	String `(auth_token pre authenticated token generated by our Systems)`

api_key

Must be sent with all client requests. The api_key helps the server to validate the request source.

auth_token

If pre authenticated method is approved then send pre authenticated token for the patron

Expected Response

Status	Response
200	`{ "auth_key": encoded String or <null> "status" : Boolean true or false }` `auth_key (string)` - all further API calls related to patron data must have this key in header
403	`{"error":"API key is missing."}`
400	`{"error":"Please provide username."}`
400	`{"error":"Please provide password."}`
401	`{"error":"Invalid API key."}`

401	`{"error":"Invalid Library Id."}`
401	`{"error":"Incorrect username or password."}`

Patron Account Information

This API pulls the Patron account information from the vendor about Patron's current checked-out items with their proposed return dates based on SLA / history of items borrowed in the past as well as any additional Patron information.

Process URL

Type	Values
GET	`https://<baseurl>/patronAccount/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
HEAD	auth_key	String (auth_key returned by login/authentication API)
POST	library_id	String
POST	selector	String (selector has 4 options 0 – All 1 – Check Out's 2 - History 3 - Holds)

Expected Response

Status	Response

200	{
	"checkout_items": array of items currently borrowed or NULL
	"history_items" : array of items returned in the current year or NULL
	"itemsOnHold" : array of items oh hold
	}
	Based on selector values return will either have All items, checkouts, history, holds

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid API key."}`

Replace Patron Barcode

This API replaces the Patron's barcode at the Vendor level. This is to address the case where a Patron is assigned a new membership card (Lost Card, Damaged Card, Stolen Card etc.). This will help us building a simple transaction at our level to reassign all content at Vendor level which was assigned to a particular patron barcode to a new barcode, allowing patron to maintain all his/her old content.

This API internally will assign all content associated with the current Patron to their new barcode.

Process URL

Type	Values
PUT	`https://<baseurl>/replaceBarcode/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	oldbarcode	String
POST	newbarcode	String

Expected Response

Status	Response

200	`{`
	`“status”: true or <false>,`
	`“response” :  {`
	`               “message” : String`
	`               }`
	`}`

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid API key."}`

Check Out Item

This API checks-out an item at the Vendor level from the items that the Library has licensed from, or hosted on, the Vendor's infrastructure. The response includes the item and its return date based on SLA. This API will be used for materials that can be downloaded.

Process URL

Type	Values
GET/POST	`https://<baseurl>/checkOutItem/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
HEAD	auth_key	String `(auth_key returned by login/authentication API)`
POST	library_id	String
POST	Item_id	String

Expected Response

Status	Response

200	`Response if check out is successful`
	`{`
	`“status”: Boolean, (true)`
	`“response” :  {`
	`              “download_url” : String,`
	`              “url_expiry_date” : timestamp, (unix timestamp)`
	`              “redownload_url” : String,`
	`              “redownload_exp” : timestamp, (unix timestamp)`
	`              “loanTimeExpires”: timestamp (unix timestamp)`
	`              }`
	`}`
	`Response if was not able to checkout`
	`{`
	`“status”: Boolean, (false),`
	`“response” :  {`
	`              “error_message” : String`
	`              }`
	`}`

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid API key."}`

Stream Audio/Video/Online Reading

This API calls the Vendor's widget to stream content that does not require inventory management (Check-out) or DRM control. This is to be used to build our discovery layer and route the user directly to streaming content so that we can provide a consistent and improved user experience.

Process URL

Type	Values
GET	`https://<baseurl>/stream/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
HEAD	auth_key	String `(auth_key returned by login/authentication API)`
POST	library_id	String
POST	Item_id	String

Expected Response

Status	Response

200	`Response if streaming is possible`
	`{`
	`“status”: Boolean, (true)`
	`“response” :  {`
	`              “streaming_widget” : String,`
	`}`
	`}`
	`Response if streaming not possible`
	`{`
	`“status”: Boolean, (false),`
	`“response” :  {`
	`              “error_message” : String`
	`              }`
	`}`

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid API key."}`

Retrieve Check-In Information

This API retrieves check-in information of Patron's items at vendor level so that we can synchronize our records with the Vendor's records. This is especially important in a situation where the vendor supports early check-in. Overnight jobs can be scheduled to get this information and update/reconcile Patron accounts in our ILS system.

Process URL

Type	Values
GET	`https://<baseurl>/ChedkInItems/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	delta_date	last run date

Expected Response

Status	Response
200	`{ “items_returned”: array object of items checked in after the given date or NULL }`
401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`

Data-store Retrieval API

This API retrieves all licensed or owned items managed by a particular vendor in order to synchronize catalog and metadata to support unified discovery and user experience.

Process URL

Type	Values
GET	`https://<baseurl>/LibraryItems/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	Itemstoreturn	String <all> or <available>
POST	date	Last run date to retrieve delta items (all items bought after this date), If Date not provided than api will return all data owned till date.

Expected Response

Status	Response

200	{
	 “items”: array object of items owned by the library and all free items
	 }
	 We expect this return array to include all meta data, proprietary Data, DRM data, any more related data, which can help us in indexing and providing bette user experience.
	 If **Itemstoreturn** is passed as “all” send all items owned by Library or if “available” is passed then array should include items owned by the library, which have items available in stock.
	 If **delta_date** is passed as the post parameter, array will include items bought after that date.

401	{"error":"Invalid Library Id."}
403	{"error":"API key is missing."}
401	{"error":"Invalid API key."}

Remove Unlicensed Items Data store API

This API identifies all items with expired licenses or items dropped from the collection being managed by the vendor. We will process this list to remove items from our catalogue or mark them to be removed form our search index

Process URL

Type	Values
GET	`https://<baseurl>/LibraryItems/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	Itemstoreturn	String <all> or <available>
POST	date	last run date to retrieve delta items (all items bought after this date)

Expected Response

Status	Response

200	`{`
	 `"items": array object of items library is discontinuing or not renewing with the vendor or just the licenses have expired`
	 `}`
	 We expect this return array to include item ids, license expiry date, reason Of non availability of item etc.
	 If **delta_date** is passed as the post parameter, array will include items expire On that date.

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`

Item Availability

This API checks current item availability of items managed by the vendor. The delta is important as it will allow us to insure that our catalog is current and synchronized with the Vendor's catalog.

Process URL

Type	Values
GET	`https://<baseurl>/availability/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	Item_id	String, blank indicates all items to be returned
POST	Date	Delta date to identify any items availability status changed since the date/time

Expected Response

Status	Response
200	`{ “availability”: array { “Total” = integer, “available” = integer “onHold” = integer } }` `Total = copies we are licensed for` `Available = number of copies available for checkout` `OnHold = number of open requests/items on hold`
401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid Item Id."}`
401	`{"error":"Invalid API key."}`

Cancel Hold/Request Item

This API removes a hold/request from the patron's account. We use this to insure that there is synchronicity between the Vendor's systems and our own.

Process URL

Type	Values
DELETE	`https://<baseurl>/hold/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
HEAD	auth_key	String (`auth_key returned by login/authentication API)`
POST	library_id	String
POST	Item_id	String

Expected Response

Status	Response
200	`{ "response": array { "status" = Boolean true or false, "error" = String } }`
401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid Item Id."}`
401	`{"error":"Invalid API key."}`

Hold/Request Item

This API initiates a hold at the Item level, if the item is currently not available.

Process URL

Type	Values
POST	`https://<baseurl>/hold/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
HEAD	auth_key	String `(auth_key returned by login/authentication API)`
POST	library_id	String
POST	Item_id	String

Expected Response

Status	Response
200	`{ “response”: array { “status” = Boolean true or false, “error” = String } }`
401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
402	`{"error":"Invalid Auth Key."}`
401	`{"error":"Invalid Item Id."}`
401	`{"error":"Invalid API key."}`

Search

This API supports those vendors which we do not catalog in ILS (often being vendors who do not provide Item-level transactions such as unlimited inventory), and allows us to integrate content from these vendors into our search experience.

Process URL

Type	Values
GET	`https://<baseurl>/search/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String

POST	page	Array of pagination options
		{
		"pagecount" : integer,
		"requestpagebyid" : integer,
		"itemsperpage" : integer
		}
		requestpagebyid : This could accept a number or a application screen "flag" name for where the user should be taken in the application or the supporting website

Continued on next page

POST	**filters**	**Expected filters for e-books and audiobooks** **Array of filters, all filters are optional** **{** **“author” : String,** **“title” : String,** **“publisher” : String,** **“audience” : String,** **“year” : integer,** **“genre” : string,** **“rating” : String,** **}** **Expected filters for Music** **Array of filters, all filters are optional** **{** **“title” : String,** **“artist” : String,** **“album” : String,** **“composer” : String,** **“director” : String,** **“genre” : String,** **“audience” : String,** **“category” : String,** **“rating” : String,** **“year” : integer** **}** **Expected filters for Movies/Videos** **Array of filters, all filters are optional** **{** **“title” : String,** **“actor” : String,** **“movie” : String,** **“producer” : String,** **“director” : String,** **“genre” : String,** **“audience” : String,** **“category” : String,** **“rating” : String,** **“year” : integer** **}** **Any other filters which are available on proprietary data. All the filters in the search are assumed optional.**

Continued on next page

Expected Response

Status	Response

200	`{`
	` “items”: array object of items owned by the library and all free items`
	`}`
	We expect this return array to include all meta data, proprietary Data, DRM data, any more related data, which can help us in providing more information the user and a better user experience.
	
	The items array should include links to download item or play/stream directly from the search results, to provide the better user experience.

401	`{"error":"Invalid Library Id."}`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`

Ranked/Featured Items

This API supports the same filters as the Search API (see above), but additionally required that we should be able to return items ranked by each filter. Examples include: 1) top n items played on a particular date, 2) top n items based on any filters we have in search.

Process URL

Type	Values
GET	`https://<baseurl>/featured/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	topItems	integer (default 20 items)

POST	page	Array of pagination options
		{
		“pagecount” : integer,
		“requestpagebyid” : integer,
		“itemsperpage” : integer
		}
		requestpagebyid : This could accept a number or a application screen “flag” name for where the user should be taken in the application or the supporting website

Continued on next page

POST	**filters**	**Expected filters for e-books and audiobooks** **Array of filters, all filters are optional** **{** **“author” : String,** **“title” : String,** **“publisher” : String,** **“audience” : String,** **“year” : integer,** **“genre” : String,** **“rating” : String,** **}** **Expected filters for Music** **Array of filters, all filters are optional** **{** **“title” : String,** **“artist” : String,** **“album” : String,** **“composer” : String,** **“director” : String,** **“genre” : String,** **“audience” : String,** **“category” : String,** **“rating” : String,** **“year” : integer** **}** **Expected filters for Movies/Videos** **Array of filters, all filters are optional** **{** **“title” : String,** **“actor” : String,** **“movie” : String,** **“producer” : String,** **“director” : String,** **“genre” : String,** **“audience” : String,** **“category” : String,** **“rating” : String,** **“year” : integer** **}** **Any other filters which are available on proprietary data. All the filters in the search are assumed optional.**

Continued on next page

Expected Response

Status	Response
200	{ "items": array object of items owned by the library and all free items } We expect this return array to include all meta data, proprietary Data, DRM data, any more related data, which can help us in providing more information t the user and a better user experience. The items array should include links to download item or play/stream directly from the search results, to provide the better user experience.
401	{"error":"Invalid Library Id."}
403	{"error":"API key is missing."}
401	{"error":"Invalid API key."}

Register Patron with Vendor

This API registers the Library Patron with a vendor. It facilitates managing barcode and password synchronization between the Library and Vendor when new patron registers at vendor level. Vendor WILL create user with barcode and pin sent by the Library and sends URL to assign new password based on vendor password policies and procedures.

Process URL

Type	Values
POST	`https://<baseurl>/patronRegister/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	patron_id	String
POST	password	String

Expected Response

Status	Response
200	`{ “status” : Boolean, true or false }`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`
401	`{"error":"Invalid Library Id."}`

Synchronize/Change Password across Vendor sites

This API allows the Library to synchronize Patron's Password at vendor level. When patron changes or resets his password with the Library, he/she will be prompted with a message, offering to automatically change their password associated with a particular vendor(s). Based upon the Patron's direction the Library either skips this change or sends new passwords to the Vendor for that Patron account based on Vendor password policies and procedures.

Process URL

Type	Values
POST	`https://<baseurl>/passwordChange/`

Parameters

Type	Parameters	Values
HEAD	api_key	String
POST	library_id	String
POST	patron_id	String
POST	password	String

Expected Response

Status	Response
200	`{ "status" : true or <false> }`
403	`{"error":"API key is missing."}`
401	`{"error":"Invalid API key."}`
401	`{"error":"Invalid Library Id."}`