NISO Flexible E-Content API Framework (FASTEN)

Introduction

In June 2016, NISO proposed a working group to build on Queens Library's work regarding establishing a worldwide conversation centered on the library industry's handling of electronic media and user interactions with this type of media, with the understanding there may be benefits and advantages for many aspects of business operations.

To date, library industry solutions had used a variety of seemingly aging, inflexible, large and hard to use enterprise tools with disparate protocols such as SIP, SIP2, proprietary interfaces, web proxy solutions, and more; without leveraging solutions that utilized the rapidly developing world of web services and mobile devices. libraries were being left behind by commercial solutions that customers were coming to expect with more dynamic and responsive feature sets that allowed for custom solutions, more flexibility, and growth for the unique and diverse markets libraries serve.

This NISO initiative took the form of having the Working Group members assess the draft of Queens Library's API document, and its proposal regarding mobile application intents and extensions that would extend interoperability for various components of integrated library systems (ILS), library discovery software, and business machines or devices.

From the outset, the group sought to modernize library-vendor technical interoperability to address the above challenges, including coming up with consistent language and data objects using RESTful web service APIs and standard mobile application intent calls.

> Using the Queens Library API Requirements as a proposed initial draft to establish and address community, vendor, and developer needs in transmitting library-related information related to serving licensed electronic content, such as:
>
> - Login/authentication
> - Account information
> - Availability
> - Item status
> - Item check-out
> - Audio/video/online recording streaming
> - Patron registration with vendor, etc.

Including creating a foundational API toolset that the library industry can build on to fulfill an array of user and library needs, including:

- Quicker response times
- Flexible item discovery and delivery options
- Improved resource availability
- More seamless integration of electronic and physical resources, plus better tracking and reporting, etc.

Also, during the exercise of reviewing the Queens Library API Draft and Mobile Intents and Extensions Proposals, the group did deep dives into

- Industry pain points
- User experience challenges
- Library business practices

that could be supported by automation, software and/or device tools, thus producing recommended practice guidance, especially in the realm of user experience for most roles and understanding of library business operations

This effort was outward looking, as well as QL draft-focused. FASTEN members had routine opportunities to add to the conversation from the depth of their experiences, be it

- Product developer
- Librarian
- Interested participant

Outward review including looping in competing, peripheral or complementary standards or working groups. Groups and standards we actively referenced are all included (e.g. BIC, LCF, [Add some more])

You will find our work separated into sections:

1. Introduction—background, goals, scope, etc.
2. Recommended practices—list of overarching principles and practices
3. Related specifications—list and context
4. API specification—This will start with a flow chart

Recommended Best Practices

Definition of Terms

Provider – is a vendor or supplier of an API
Customer – is the Customer of the Provider
User – are the Users of the Customer

Resource – are the products of the Provider (ebooks etc.)
ILS - Integrated Library System
SSO - single sign-on

For line ref(s), see the table in Appendix A which is a copy of items listed as ‘Recommended Practices’ from the ‘Best Practice Review’ spreadsheet.

Implementation of API

Line ref(s). 1,2,3,34,39

When designing an API for use, considerations should be made as to how it will be applied/used by the customer. Likewise, the customer has responsibilities to use the API in a way that doesn’t detract from its purpose. In particular the following considerations should be looked at:

Path to resource (1)

The provider of a resource should design an API that allows the users of the customer’s LMS to remain in their native environment without having to follow multiple steps in order to access it. The journey of the user to the resource should be intuitive and, where possible, seamless without the need for separate or repeated logins. The user should not be taken away to the provider’s interface as this is disruptive to the user.

Appropriate design and interface design (2,3)

The customer should display data from or integrate the API in a way which is consistent with the resource. For example, it would be inappropriate to have media buttons (such as play, stop, etc.) to control an ebook or to require “hover” functionality for mobile devices.

Whilst providers should include descriptors to help customers determine the best way to implement the API. For example, if the API informs the customer that the resource they are getting is an ebook, then this allows the customer to design appropriately.

Handling multiple sources (34)

Customers should aim to incorporate all matching search results in to single results interface to present the user with a comprehensive list of matching resources.

Handling multiple sources will mean that providers should enable mechanisms to provide graceful failure to allow customer systems to continue even if resources from a particular API are unavailable.

Resource information (39)

Where information or content regarding a resource is available on a providers platform, then this should be available to the customer in the form of an API. For example where resources, including the formation of categories, collections or lists, is being paid for by the customer, then these elements, alongside the resource, should be available for integration in to the customer's platform.

Authenticating Users and Personal Data

Line ref(s). 5,21,38,46,48

Sign-up (5)

The specifics of authentication is beyond the scope of these recommendations but consideration should be given to how a User will be authenticated at the Customer (or ILS) level and then to be authorized at the Provider level. In particular the steps taken by the User should not be excessive and limits should be placed upon how long it takes, both for sign-up and to establish identity.

Synchronisation of patron details/password (46 & 48)

With increasing moves towards SSO as a point of access, interoperability with such mechanisms needs to be carefully monitored as will impact upon various workflows such as password synchronisation (46) or updating of patron details (e.g. barcode) (48).

Handling personal data and consistency of experience (21 & 38)

The handling and security of personal User data is of paramount importance. Personal information should not be passed between Customer and Provider, authentication of the User should happen as a User > Customer interaction, whilst authorization to access a particular resource should occur as part of a Provider > Customer interaction as this limits the exposure of personal information (21). Keeping the User in a library branded interface whilst serving Resources (38) implies that the User > Customer interaction is the only one taking place, where User information is passed on to the Provider, the User should be informed.

Finding and Using Content

Line ref(s). 15,28,39,42,43

Checkout at Title Level (15)

From the Library Business document:

> Checkouts should be at the title level, not at the format level. That is, checking out a title should give access to all formats in which that title is available. (Within a type, of course: this doesn't apply to video vs text but does apply to Kindle vs ePub.)

From an Academic perspective, we can have the same title of an ebook but multiple access rights from different publishers.

Streaming Content (43)

When streaming content that does not require DRM or check-out, patrons have a seamless experience so they instantly access the streamed content and don't have to register for accounts or install additional software or apps for streaming.

Access to streaming resources that aren't secured by DRM or that require a check-out, should occur in the library environment

Display of Content (28, 39 & 42)

As well as supplying information about a given resource, e.g. title and format etc., additional data should be supplied that allows for appropriate use of the records or to maintain existing links with grouped records. For instance, when supplying information regarding items in a curated list or collection, then this information should be included alongside the item information (39). By doing so these collections are preserved outside of vendors proprietary systems keeping the library user in the library domain.

In addition, where items have specific constraints placed upon them, such as licence rights, then this should be returned with the metadata. This is also important when considering consortia, having information about which libraries have access to the item would be necessary for discovery (42).

Having controls over which titles can be imported or found via a search is necessary for implementation (28) as this allows discovery tools to filter out items based on a particular category or level.

Functional Considerations

Line ref(s). 13,29,30,32,38,46

Supply of these APIs to Customers, requires certain technical or functional considerations. The purpose of these considerations is to help the Customer implement them whilst striving to reduce elements of doubt about their technical capability. For instance clarity over whether the endpoint is suitable for their needs, provision of a testing API and dealing with changes/new releases.

Separate user endpoints (13)

Separate logins for the three types of system users: Customers/Users, Library Admins, Content Providers. See Best Practices Breakdown of UX Subgroup
Is this a spec or recommended practice? Needs more work logins, or permission, or different UI levels - are API elements in here? API practice. Relationship with library business

Provision of a testing/sandbox API (29)

With a each production API provided, it will be necessary to provide a testing or sandbox API that exactly replicates the production API in terms of content and functionality. This allows for accurate testing by the Customer and establishes what they need to do in order to implement it fully. Having an exact copy of the production API in terms of content and functionality further removes elements of doubt that might surface due to testing with different records

Changes to an endpoint (30)

Before a Provider of an API makes changes to the endpoint, they should communicate, well in advance, the nature of the changes and, for a reasonable time, allow both old and new endpoints to run simultaneously before removing the old one(s).

Help Pages and Error Handling

Line ref(s). 4,11,12

Graceful in failure (4,11 & 12)

When building help around APIs or around services that make use of delivered APIs, the user experience should be graceful in failure providing descriptive help, outlining where an error

occurred and possible reasons why it did. Help should be descriptive for developers but also offer a meaningful message to the user. They should build upon the wide range of descriptive error messages that are available whilst messages designed for end users should provide information on how to recover from the problem (if possible), or provide contact information for assistance.

Avoid enormous “select your library” pulldowns

Traditionally, eContent vendor apps must, at least one time, make the patron select his library in order to log in. Often this involves a large list of all libraries with which the eContent vendor has a relationship. GPS location or prompts for ZIP codes can help shorten the list, although that requires yet another prompt.

If the patron starts with the eContent app, this selection is probably unavoidable. However, if the patron starts with the ILS, then there are ways to reduce the inconvenience.

If the app resides on the same device that is used for ILS login, none of this may be necessary.

The eContent vendor should generate a code of approximately six characters, either when a patron logs in or when information about connecting an app is requested. The code should only be valid for a relatively brief length of time (probably no more than an hour).

This code can be entered into the app, and will associate the app with the correct library. It may also log the patron in. This might not be the only way the patron can log in to the app: if the patron starts at the app rather than the ILS next time, the app can handle the login, but will be able to skip the "select your library" step.

[For the case where the FASTEN API is used for checking out and fulfilling (as opposed to sending the patron to the eContent web site), but there still is an associated app run by the eContent vendor, should there be a way for the ILS to request a code to give the user?]

Appendix A – Items from Best Practice Review

Key

Library UX

Library Business

Library Discovery and Search

1	Path to Resource	Path from finding a resource to accessing it should be intuitive, seamless and offer few/no barriers to the end user. The user should not have to leave the environment in which the resource was found. The resource should be displayed within the library catalogue/discovery layer and should not be taken to a third party website, e.g., publisher or aggregator site.	
2	Appropriate functionality	UX should incorporate functionality that is standard for that environment, e.g., do not use play buttons to indicate next page in an ebook, do not require “hover” functionality for mobile devices.	
3	Interface Design	Intuitive, clean, consistent interface design. Use familiar, widely implemented control features when possible (hamburger icon, play button, etc). Search and Discovery should be easy to use. User should organically recognize they are in “browse” mode (with the ability to sort or filter media on screen) and also initiate keyword/advance searching informed by visual cues in the interface.play media	
4	Well Designed Help	Easy to navigate help (if initial contextual help/error assistance is insufficient).	
5	User Sign-up	Sign-up, should be quick, in under two minutes to complete, and max five steps (screens) minimum data points to establish identity and membership to a community.	
6	DRM Authentication	Media should allow for Library/Open Source readers that allow DRM authentication	

11	Fail Gracefully	The user experience should be graceful in failure, providing an understandable message to the user and not just status codes or internal error messages. APIs should make use of the wide range of statuses to inform the application making use of them: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html. Messages for the user should acknowledge that there will be problems and help them successfully recover from the problem.	
12	Contextual Help	Contextual help and assistance for errors.	
13	Separate User Endpoints	Separate logins for the three types of system users: Customers/Users, Library Admins, Content Providers. See Best Practices Breakdown of UX Subgroup	Is this a spec or recommended practice? Needs more work logins, or permission, or different UI levels - are API elements in here? API practice. Relationship with library business
15	Checkouts should apply at the title level, and any/all file types associated with that title should be included		
21	Personal patron information	No personal patron information should move from the identity provider (or ILS or discovery layer) to a third-party eContent vendor. That way there's no potential for abuse.	Might be recommended standard (this seperation of authentication and authorisation is part of shib for example). This is a principle that needs to underpin the other authentication work

28	Filter titles	The ILS should allow certain titles in eContent libraries to be ignored (not imported and/or not found via search)	
29	#3 Testing/sandbox API not matching live API	All parties (vendors, customers, third-party intermediaries) need to provide testing/sandbox/QA APIs and access to data that matches production content and functionality.	
30	#5 Supplier/vendor changes the API endpoint	Any API endpoint change should: 1) be communicated to users of the API well in advance of the change; 2) allow both the old and new endpoints to work simultaneously for a specified reasonable amount of time before removing the old one(s).	
32	#4 Lack of backwards compatibility when a new version is released	If at all possible, new versions of software should include, minimally, the functionality of the version it is replacing for a reasonable amount of time.	
34	#12 Searching assumptions for ILS Catalog (loaded into discovery or federated search)	Surfacing all matching results in a single interface; having all metadata available to customize as desired. This front-end simplicity for the user may require backend complexity, but this simplexity is critical, and additional technical work by the library is acceptable as long as vendors provide a standardized way to do this (with APIs).	
38	#17 Lack of “web awareness” of current legacy ILS; i.e. the public OPAC’s robot.txt restricting indexing by Google or Bing and other web indexers	As much as possible, the patrons should be using or taken to a library-branded interface so they are not concerned with privacy or quality issues. Library applications should have specialized SEO APIs that can be used to generate robots.txt and meta tags for increased SEO friendliness of OPAC and Discovery layers.	
39	#25 If an e-resource is coming from a curated collection, information about the curation should not be lost to the catalog display.	If we are paying for content (including metadata) available on the vendor’s proprietary site, we should be able to move that to our portal. For example, BiblioBoard provides categories and Freegal provides lists such as "top 10 artists." Both of these are not available outside their proprietary systems.	may require hook api spec.

42	#28 Consortial scenarios for eBook access need to be considered	As a patron using a shared catalog, I want to easily determine which e-books are available to me so that I can check them out through the catalog or other client. As a consortium manager, I want to show e-book titles from multiple sources and with varying license rights in the shared catalog so that patrons can easily find which e-books are available to them. For example, the same e-book title may be owned by different libraries through different e-book consortia using different vendors and may also be owned by some libraries through individual library purchases. As a consortium manager, I want to be able to sort and prioritize sources of ebooks by library so I can spend money efficiently and effectively.	
43	Queens: Stream Audio/Video/Online Reading	When streaming content that does not require DRM or check-out, patrons have a seamless experience so they instantly access the streamed content and don't have to register for accounts or install additional software or apps for streaming.	Shouldn't really need a new API but might!
46	Queens: Synchronize/Change Password across Vendor sites	Single sign on may not be a best practice in practice, because it's so difficult to achieve, but it is what technology has been moving toward for many years.	Various technologies exist for password sync (e.g. basic ldap) - related the other sso topics
48	Queens: Replace Patron Barcode	APIs should make the transition from one card to another (assuring security in case of stolen cards) seamless, allowing content from an earlier library barcode to remain in the account even if the barcode is updated.	This sounds like a design decision - e.g. the underlying database should not use barcode as a immutable primary key

Accompanying Standards

Resource Synchronization

There are two fundamental options for how to include eContent records in ILS/Discovery search results:
A) The ILS/Discovery layer ingests MARC provided by the eContent vendor.
B) For each appropriate user search, the ILS/Discovery layer calls a search API provided by the eContent vendor and includes eContent results alongside its native results.

In FASTEN, the approach to use is the one mutually agreed upon between the ILS and the eContent vendor. It seems as of this writing that the best practice is to use method A when practical, and that method A becomes impractical when eContent collections are prohibitively large.

Method A - MARC ingestion:
The protocol chosen for keeping the ILS MARC in sync with the eContent vendor's MARC should meet the following requirements:
* An open, published, non-proprietary standard.
* Baseline synchronization - the ILS must be able to perform an initial load or catch-up with a source at any time.
* Incremental synchronization - the ILS must be able to request and receive a list of changes since the last synchronization.
* Audit - the ILS must be able to verify that its current collection matcthes the eContent vendor's collection.

The recommended mechanism for Method A at this time, particularly for new implementations, is NISO ResourceSync. OAI-PMH also satisfies these requirements.

Method B - Search API:
The protocol chosen for real-time programmatic searching of the eContent vendor's database should also be an open, published, non-proprietary standard. Current good options are:
* Z39.50
* SRU
* SRW
The current recommendation for new implementations is SRU: unlike Z39.50, it works on top of HTTP, so the lower-level protocol is better-supported by software libraries and firewalls; and it is simpler than the SOAP-based SRW.

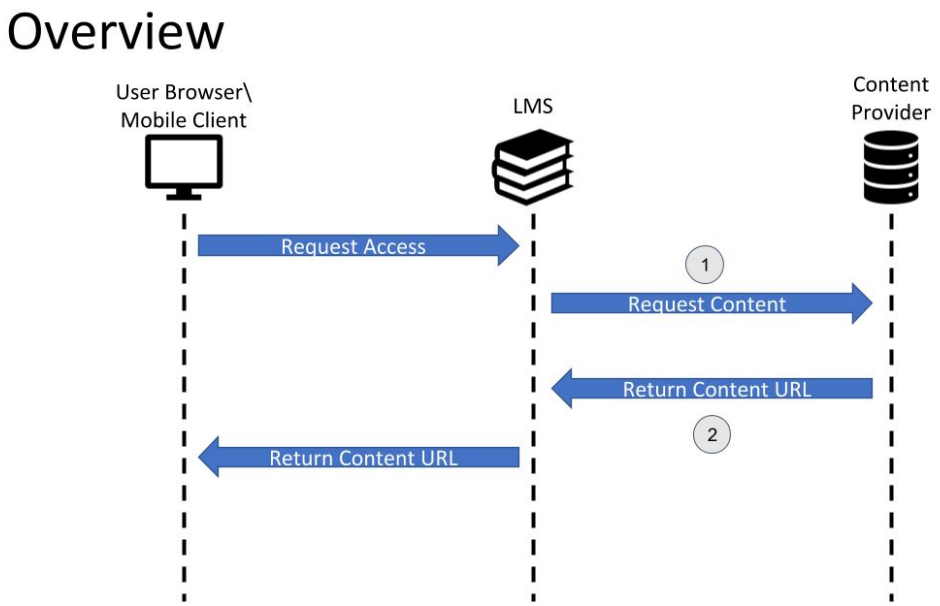
[mention/require Bath Profile or similar?]

There are many issues with attempting to map a patron's search into a query that makes sense to a vendor's Z39.50/SRU/SRW server. FASTEN is unfortunately not able to specify solutions for all those issues. Future versions of this document may include more specific best practices.

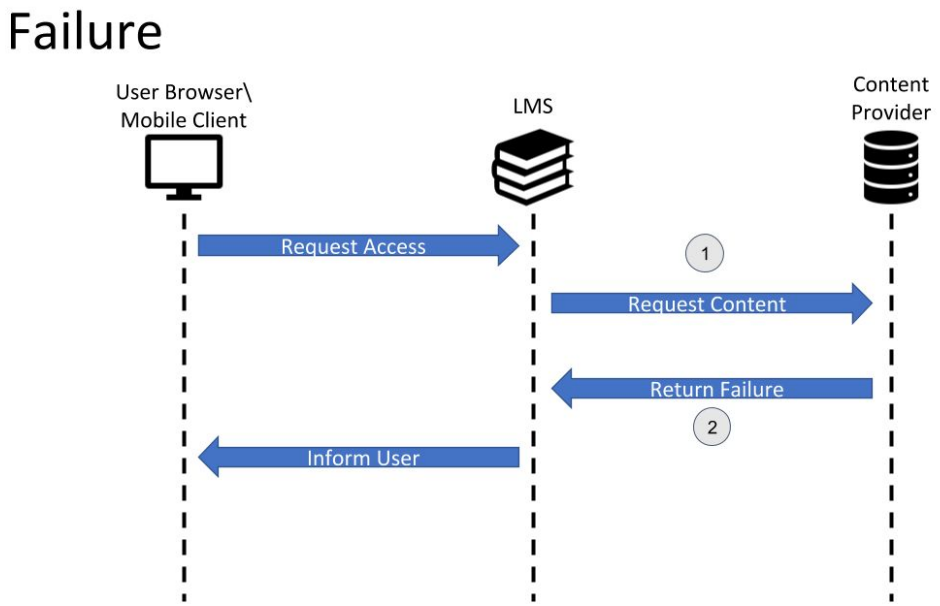
FASTEN NCIP Implementation Profile

Supported Scenarios

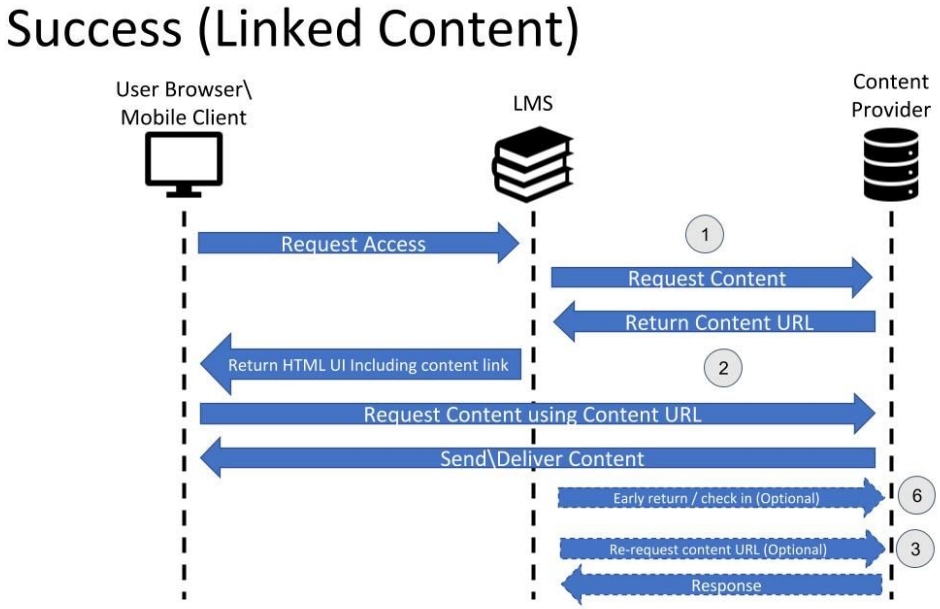
Overview



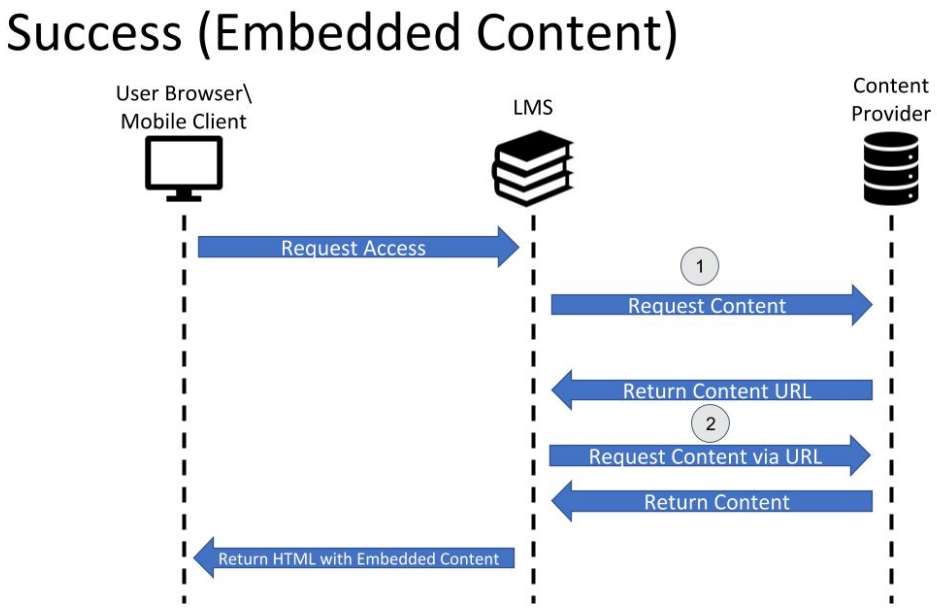
Failure Mode



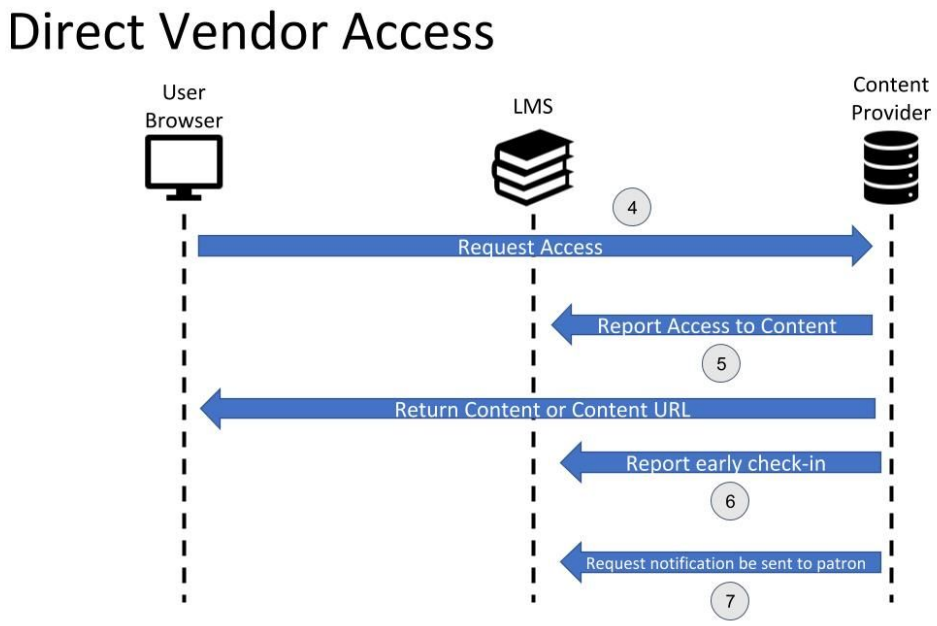
Success - Linked Content



Success - Embedded Content



Director Vendor Access (Vendor to ILS communication)



NCIP and LCF Messages

Request #	NCIP Message	LCF Message
1	CheckOutItem	Create Loan Request
2	CheckOutItemResponse	Create Loan Response
3	LookupItem	Update Loan Request and Response
~~4~~	~~?? CheckOutItem & CheckOutItemResponse~~	~~?? Create Loan Request & Create Loan Response~~
5	ItemCheckedOut	Update Loan Request and Response
6	CheckInItem	Check In Loan Request and Response
7	SendUserNotice	Create message-alert

NCIP Messages

RequestItem

The `RequestItem` message is used by the ILS to request that the e-content vendor add the specified patron to a wait list for the specified, currently unavailable, item.

The following fields are required:

- From Agency ID (representing the library)
- User ID
- Item ID

Sample XML:

```
<?xml version="1.0"?>
<ncip:RequestItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
    <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:ItemId>
    <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
</ncip:RequestItem>
```

RequestItemResponse

The `RequestItemResponse` message is used by the eContent vendor to confirm that the patron has been added to the the e-content vendor add the specified patron to a wait list for the specified, currently unavailable, item.

The following fields are required:

- From Agency ID (representing the vendor)
- Request ID

Sample XML:

```
<?xml version="1.0"?>
<ncip:RequestItemResponse xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
```

```
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:RequestId>
    <ncip:RequestIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:RequestId>
</ncip:RequestItemResponse>
```

CancelRequestItem

The `CancelRequestItem` message is used by the ILS to request that the e-content vendor cancel a previous request.

The following fields are required:

- From Agency ID (representing the library)
- Request ID
- [NCIP seems to require User ID.. Should we require it, or require that it be present but empty, or…?]

Sample XML:

```
<?xml version="1.0"?>
<ncip:CancelRequestItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:RequestId>
    <ncip:RequestIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:RequestId>
</ncip:CheckOutItem>
```

CheckOutItem

The `CheckOutItem` message is used by the ILS to request that the e-content vendor provide the requested item to the specified patron.

The following fields are required:

- From Agency ID (representing the library)
- User ID
- Item ID

Sample XML:

```
<?xml version="1.0"?>
<ncip:CheckOutItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
```

```
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
    <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:ItemId>
    <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
</ncip:CheckOutItem>
```

CheckOutItemResponse

The `CheckOutItemResponse` is used by the e-content vendor to confirm that the item has been charged to the specified user and to provide URL by which the patron can gain access to the content.

The following fields are required:

- From agency ID (representing the vendor)
- Problem (or)
- User ID (required by NCIP but ignored)
- Item ID (required by NCIP but ignored)
- Due date
- Reference to Resource
- Loan ID (in an <Ext/>)
- Additional Reference(s) to Resource (if needed, in an <Ext/>)

Sample XML:

```
<?xml version="1.0"?>
<ncip:CheckOutItemResponse xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>

<!-- EITHER -->

  <ncip:Problem>
    <ncip:ProblemType ncip:Scheme="">string</ncip:ProblemType>
      <!--Optional:-->
      <ncip:ProblemDetail>string</ncip:ProblemDetail>
      <!--Optional:-->
      <ncip:ProblemElement>string</ncip:ProblemElement>
      <!--Optional:-->
      <ncip:ProblemValue>string</ncip:ProblemValue>
  </ncip:Problem>

<!-- OR -->

  <ncip:ItemId>
```

```
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <ncip:DateDue>2017-08-18T22:01:40</ncip:DateDue>
  <ncip:ElectronicResource>
    <!--EITHER-->
    <ncip:ElectronicDataFormatType ncip:Scheme="">string</ncip:ElectronicDataFormatType>
    <ncip:ActualResource>string</ncip:ActualResource>
    <!-- OR -->
    <ncip:ReferenceToResource>string</ncip:ReferenceToResource>
    <!-- END CONDITIONAL -->

    <!-- Specify the destination type -->
    <ncip:Ext>
        <ns2:ElectronicResourceUsageType
ncip:Scheme="http://niso.org/fasten/electronicresourceusagetype/fastenusagetypes.scm">string</
ns2:ElectronicResourceUsageType>
    </ncip:Ext>
  </ncip:ElectronicResource>

  <ncip:Ext>
    <ns2:LoanId>
      <ns2:LoanIdentifierValue>string</ns2:LoanIdentifierValue>
    </ns2:LoadId>
  </ncip:Ext>

  <!-- If there are multiple resources to be returned, additional resources are included thus:
-->
    <ncip:Ext>
        <ncip:ElectronicResource>
            <!--EITHER-->
            <ncip:ElectronicDataFormatType
ncip:Scheme="">string</ncip:ElectronicDataFormatType>
            <ncip:ActualResource>string</ncip:ActualResource>
            <!-- OR -->
            <ncip:ReferenceToResource>string</ncip:ReferenceToResource>
            <!-- END CONDITIONAL -->

            <!-- Specify the destination type -->
            <ncip:Ext>
                <ns2:ElectronicResourceUsageType
ncip:Scheme="http://niso.org/fasten/electronicresourceusagetype/fastenusagetypes.scm">string</
ns2:ElectronicResourceUsageType>
            </ncip:Ext>
        </ncip:ElectronicResource>
    </ncip:Ext>

</ncip:CheckOutItemResponse>
```

Allowed values for “ElectronicResourceUsageType” are “browser_player” and “raw_media”.

ItemCheckedOut

The `ItemCheckedOut` message is optionally used by an e-content vendor to report that a resource was directly requested and charged to a patron. In this case, the e-content vendor receives a request for a resource which is considered valid according to the authentication scheme previously defined between the library and the e-content vendor. This may include a properly signed token (see Authentication below).

The following fields are required:

- From agency ID (representing the vendor)
- User ID (derived from the request)
- Item ID
- Due Date (optional if content is returned automatically after a certain period)

Sample XML:

```
<?xml version="1.0"?>
<ncip:ItemCheckedOut xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
    <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:ItemId>
    <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
 <ncip:DateDue>2017-08-18T22:01:40</ncip:DateDue>
</ncip:ItemCheckedOut>
```

ItemCheckedIn

The `ItemCheckedIn` message is optionally used by an e-content vendor to report that a resource was returned by the patron. In this case, the e-content vendor receives a request to return an item and reports that return to the ILS.

The following fields are required:

- From agency ID (representing the vendor)
- User ID (derived from the request - required by NCIP but ignored)
- Item ID (required by NCIP but ignored)
- Loan ID (in an <Ext/>)

Sample XML:

```
<?xml version="1.0"?>
<ncip:ItemCheckedIn xmlns:ncip="http://www.niso.org/2008/ncip">
```

```
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
    <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:ItemId>
    <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <ncip:Ext>
    <ns2:LoanId>
      <ns2:LoanIdentifierValue>string</ns2:LoanIdentifierValue>
    </ns2:LoadId>
  </ncip:Ext>
</ncip:ItemCheckedIn>
```

CheckInItem

The `CheckInItem` request is optionally used to return an item before its due date. The e-content vendor will return the specified item.

The following fields are required:

- From Agency ID (representing the library)
- Item ID (required by NCIP but ignored)
- Loan ID (in an <Ext/>)

Sample XML:

```
<?xml version="1.0"?>
<ncip:CheckInItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
    <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
    </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:ItemId>
    <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
    <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <ncip:Ext>
    <ns2:LoanId>
      <ns2:LoanIdentifierValue>string</ns2:LoanIdentifierValue>
    </ns2:LoadId>
  </ncip:Ext>
</ncip:CheckInItem>
```

RenewItem

The `RenewItem` request can be used by an ILS to request that a checkout be renewed.

The following fields are required:

- User ID (required by NCIP but ignored)
- Item ID (required by NCIP but ignored)
- Loan ID (in an <Ext/>)

Sample XML:

```
<?xml version="1.0"?>
<ncip:RenewItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
       <ncip:FromAgencyId>
       <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
       </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
    <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:ItemId>
       <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
       <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <ncip:Ext>
    <ns2:LoanId>
      <ns2:LoanIdentifierValue>string</ns2:LoanIdentifierValue>
    </ns2:LoadId>
  </ncip:Ext>
</ncip:RenewItem>
```

LookupItem

The `LookupItem` request can be used by an ILS to retrieve an updated content URL. This can be useful in a scenario where the content vendor provides a URL during the check out event which is temporal and must be refreshed.

The following fields are required:

- Item ID (required by NCIP but ignored)
- Loan ID (in an <Ext/>)
- Request the Electronic resource fields

Sample XML:

```
<?xml version="1.0"?>
<ncip:LookupItem xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
       <ncip:FromAgencyId>
       <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
```

```
      </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:ItemId>
      <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
      <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <!--Zero or more repetitions:-->
  <ncip:ItemElementType ncip:Scheme="">ElectronicResource</ncip:ItemElementType>
  <ncip:Ext>
    <ns2:LoanId>
      <ns2:LoanIdentifierValue>string</ns2:LoanIdentifierValue>
    </ns2:LoadId>
  </ncip:Ext>
</ncip:LookupItem>
```

LookupItem Response

Required fields:

- Actual resource or reference to resource fields with the URL

Sample XML:

```
<?xml version="1.0"?>
<ncip:LookupItemResponse xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:ResponseHeader>
      <ncip:FromAgencyId>
      <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
      </ncip:FromAgencyId>
  </ncip:ResponseHeader>
  <ncip:ItemId>
      <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
      <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
  </ncip:ItemId>
  <ncip:ItemOptionalFields>
      <ncip:ElectronicResource>
      <ncip:ElectronicDataFormatType ncip:Scheme="">string</ncip:ElectronicDataFormatType>
     <!--- EITHER --->
      <ncip:ActualResource>string</ncip:ActualResource>
     <!--- OR ---->
      <ncip:ReferenceToResource>string</ncip:ReferenceToResource>
      </ncip:ElectronicResource>
  </ncip:ItemOptionalFields>
</ncip:LookupItemResponse>
```

SendUserNotice

The `SendUserNotice` request can be used by content vendor to request that the ILS send a notification to a patron on its behalf.

The following fields are required:

- User ID
- Notification text/event to be sent

```
<?xml version="1.0"?>
<ncip:SendUserNotice xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:InitiationHeader>
       <ncip:FromAgencyId>
       <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
       </ncip:FromAgencyId>
  </ncip:InitiationHeader>
  <ncip:UserId>
       <ncip:UserIdentifierType ncip:Scheme="">string</ncip:UserIdentifierType>
       <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <ncip:DateToSend>2009-11-03T13:39:15+01:00</ncip:DateToSend>
  <ncip:UserNoticeDetails>
       <ncip:NoticeType ncip:Scheme="">string</ncip:NoticeType>
       <ncip:NoticeContent>string</ncip:NoticeContent>
       <ncip:NoticeItem>
       <ncip:ItemDetails>
       <ncip:ItemId>
       <ncip:ItemIdentifierType ncip:Scheme="">string</ncip:ItemIdentifierType>
       <ncip:ItemIdentifierValue>string</ncip:ItemIdentifierValue>
       </ncip:ItemId>
       </ncip:ItemDetails>
  </ncip:UserNoticeDetails>
</ncip:SendUserNotice>
```

SendUserNoticeResponse

The ILS can respond with the result of the request.

```
<?xml version="1.0"?>
<ncip:SendUserNoticeResponse xmlns:ncip="http://www.niso.org/2008/ncip">
  <ncip:ResponseHeader>
       <ncip:FromAgencyId>
       <ncip:AgencyId ncip:Scheme="">string</ncip:AgencyId>
       </ncip:FromAgencyId>
  </ncip:ResponseHeader>
  <ncip:UserId>
       <ncip:UserIdentifierType ncip:Scheme="">string</ncip:UserIdentifierType>
       <ncip:UserIdentifierValue>string</ncip:UserIdentifierValue>
  </ncip:UserId>
  <!--You have a CHOICE of the next 2 items at this level-->
  <ncip:DateSent>2002-11-16T10:03:57+01:00</ncip:DateSent>
  <ncip:DateWillSend>2000-09-14T10:07:34</ncip:DateWillSend>
</ncip:SendUserNoticeResponse>
```

LCF Messages

Create Loan

POST to https://content.server/lcf/1.0/loans

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref>
  <start-date>checkout-date</start-date>
  <loan-status>01</loan-status>
</loan>
```

Create Loan Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <identifier>loan-id</identifier>
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref>
  <start-date>checkout-date</start-date>
  <end-due-date>due-date</end-due-date>
  <access-link>
    <link-type>01</link-type> <!- Direct link to resource ->
    <link>http://content.server/content.pdf</access-link>
  </access-link>
  <access-link>
    <link-type>02</link-type> <!- Indirect link to resource ->
    <link>http://content.server/viewer/content</link>
  </access-link>
  <loan-status>01</loan-status> <!—checked-out-->
</loan>
```

Update Loan Request and Response

POST to https://library.server/lcf/1.0/loans (for initial update)
PUT to https://library.server/lcf/1.0/loans/loan-id (for subsequent updates)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

```
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref> <!-- library
system uses content.server item-id to locate item\patron -->
  <start-date>checkout-date</start-date>
  <end-due-date>due-date</end-due-date>
  <access-link>
    <link-type>01</link-type> <!- Direct link to resource ->
    <link>http://content.server/content.pdf#PositionLocationID</link>
  </access-link>
  <access-link>
    <link-type>02</link-type> <!- Indirect link to resource ->
    <link>http://content.server/viewer/content#PositionLocationId</link>
  </access-link>
  <loan-status>01</loan-status><!—checked-out but could use other codes, for
failure etc.-->
</loan>
```

Response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <identifier>loan-id</identifier>
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref> <!-- library
system uses content.server item-id to locate item\patron -->
  <start-date>checkout-date</start-date>
  <end-due-date>due-date</end-due-date>
  <access-link>
    <link-type>01</link-type> <!- Direct link to resource ->
    <link>http://content.server/content.pdf#PositionLocationID</link>
  </access-link>
  <access-link>
    <link-type>02</link-type> <!- Indirect link to resource ->
    <link>http://content.server/viewer/content#PositionLocationId</link>
  </access-link>
  <loan-status>01</loan-status><!—checked-out but could use other codes, for
failure etc.-->
</loan>
```

Check In Loan Request and Response

Request

PUT to https://content.server/lcf/1.0/loans/**loan-id**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <identifier>loan-id</identifier>
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref>
  <start-date>checkout-date</start-date>
  <loan-status>08</loan-status><!-- checked in -->
</loan>
```

Response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<loan xmlns="http://ns.bic.org.uk/lcf/1.0"
xmlns:ns2="http://a9.com/-/spec/opensearch/1.1/">
  <identifier>loan-id</identifier>
  <patron-ref>https://lms.server/lcf/1.0/patrons/patron-id</patron-ref>
  <item-ref>https://content.server/lcf/1.0/items/item-id<item-ref>
  <start-date>checkout-date</start-date>
  <end-due-date>due-date</end-due-date>
  <loan-status>08</loan-status><!-- checked in -->
</loan>
```

Create Message-Alerts

POST to https://library.server/lcf/1.0/message-alerts

```
<message-alert xmlns="http://ns.bic.org/lcf/1.0">
<!-- optional fields: priority display-type display-constraint
start-date end-date -->
  <message-type>01</message-type> <!-- Action required -->
  <message-text>
     <message-format>03</message-format> <!-- Unicode -->
     <text>Message</text>
  </message-text>
  <audience>02</audience>  <!-- Specified patron and categories -->
  <patron-ref>...</patron-ref>
  <patron-ref>...</patron-ref>
  <patron-ref>...</patron-ref>
</message-alert>
```

Response

```
<message-alert xmlns="http://ns.bic.org/lcf/1.0">
  <identifer>...</identifier>
  <message-type>01</message-type> <!-- Action required -->
```

```
  <message-text>
     <message-format>03</message-format> <!-- Unicode -->
     <text>Message</text>
  </message-text>
  <audience>02</audience>  <!-- Specified patron and categories -->
  <patron-ref>...</patron-ref>
  <patron-ref>...</patron-ref>
  <patron-ref>...</patron-ref>
</message-alert>
```

Issue to raise with LCF Technical Committee - how to confirm delivery

Authentication

Vendor-to-Vendor Authentication

Clients should authenticate servers by trusting the TLS certificate they present (most likely, this will be a typical CA certificate as commonly used on the public Web). Servers should authenticate clients by validating a username and password included in HTTP Basic Authentication or by verifying that the client possesses a client certificate which has been signed by the server. Either Basic Authentication or the client certificate must be included on each request.

Patron Authentication Introduction and Principles

FASTEN aims for:

- Anonymity — eContent vendors are given as little information as possible about library patrons.
- Single sign on (if signing on is necessary) — the library's ILS (or other identity provider) is the sole repository of account information, and patrons' usernames and passwords are only ever sent to the identity provider, never to any other vendor.
- Confidentiality — Nothing in any way related to patron data ever crosses the Internet unencrypted.

Single Sign On (if necessary)

In FASTEN, patrons have only one account: the account at their library's identity provider (most likely, the ILS). They do not have accounts at eContent vendors.

If both sides fully implement FASTEN, then it's possible for the patron to only interact with the ILS and never the eContent vendor directly. This situation is best for privacy: the ILS tells the

eContent vendor what to check in and out, but no information whatsoever about any patron is needed to be sent at all.

The more common scenario (at least currently) is that the eContent vendor has its own site, or app, or both, which needs to authenticate patrons. In this situation, Open ID Connect is the FASTEN mechanism for achieving single sign on. Because logging out is an important consideration in this context, the optional Open ID Connect session management and logout specifications should also be implemented. (This prevents a patron from logging into an eContent vendor, then logging out, and not realizing that there's still an active session with the ILS.)

Anonymity

The identity provider (ILS) should send no information about patrons to the eContent vendor with the exception of an identifier (and even then only when necessary). This identifier:

- Should not be based on a barcode, which could change
- Should be pseudonymized — an identifier sent to a vendor should be meaningless in any other context, including other eContent vendors which the ILS integrates with via FASTEN
- May be fully anonymized — if the ILS/identity provider (or librarian using it) wishes patrons to be fully anonymized, then a new pseudonymized identifier can be generated for a given patron each time that patron logs in. This prevents the eContent vendor from knowing anything about any patron across different login sessions. It may harm usability, in that the eContent vendor's website, app, etc, will not be able to list all the items a patron has checked out.
- May be completely unnecessary - If all interaction with the eContent vendor takes place through the ILS, then the eContent vendor never needs to see a patron ID at all.

Confidentiality

All messages must be conducted via HTTPS. Some software libraries default to hostname verification being turned off; it (along with all the other verifications such as expiration date, etc) must be enabled in production.

Open Issues

Issue #	Date	Summary	Notes	Status
1	9-May	Consortial identification	How do we represent institution and member	Closed- authentication is up to the library and the vendor. Fields within the messages can be used.
2	9-May	Can we return both a link to the content and a link to the viewer	XSD shows either ReferenceToResource or ActualResource Xan suggests to allow both- perhaps we can raise this to NCIP (would be backwards compatible).	Closed - LCF supports natively. For NCIP, we're using the <Ext/> extensibility.
3	11-Jul	Synching catalog data	Do we recommend ResourceSync or require it? Require that the IDs necessary to complete the NCIP calls be shared between the ILS and the vendor. Also, we may need to specify an API for live-searching records as opposed to ingesting them. ResourceSync or OAI/PMH for just-in-case synchronization of the catalog or SRU for just-in-time searching. To be added to document.	Closed
4	11-Jul	LCF vs NCIP	Need to finalize on the protocol NCIP Pros- NISO standard, more widely adopted by vendors/ILS systems (albeit other implementation profiles) LCF Pros- Native support for additional fields (content URL, loan ID)	Closed

			LCF seems to be a better fit but is too new. Need to explore if we can adapt NCIP to an "LCF model"- working with loan IDs which will make it more digital-friendly. **Decision**: Since implementing this standard will require work regardless, let's go with the more forward looking approach and prescribe LCF. Next steps: Add a paragraph describing why we decided LCF and move the NCIP implementation to an Appendix. (**Josh**)	
5	8-Aug	NCIP Electronic Resource Types	Finalize what they are (do we need more than two?) and figure out how to make them official -- wrote to John Bodfish @ OCLC	Closed- NCIP extension names above.
6	8-Aug	Include MIME Types	Do we want to include the MIME type of a resource URL's destination wherever such a URL is given?	Closed- not needed.
7	8-Aug	Notifications	There needs to be a mechanism for an eContent vendor to trigger the ILS to send notifications to patrons. Perhaps the ILS could poll the eContent vendor rather than establishing a connection the other direction. In any case, we need an API for this. For example- your electronic item is available. Next steps- identify LCF (**Matthew**) API for this. - Done	Open

			Josh- Add to specification and mark as optional- validate the optional approach works for vendor to ILS communication (for checkout and early checkin too) (Added early checkin, send notification, and checkout as vendor-to-ILS communication.) TODO: probably want to specify a vocabulary of message types rather than message text? (**Matthew**)	
8	8-Aug	Inform ILS about early return	We say that the eContent vendor should tell the ILS about checkouts. What about checkins? In a situation where there's full anonymity and no LCF-style loan IDs, this could be critical: the only way for the patron's "items out" list to be correct. Otherwise it may be less critical, because the ILS could look up current information when it needed to. In the case of early return, confirm with e-content vendor before using checkout information (notifications, etc.) Next step: default position is to recommend the ILS validate checkouts. Consider if a way to poll the e-content vendor is possible/desirable. See above #7 (**Josh**) Added optional check in from vendor to ILS	Open

			TODO: Define the message in LCF as optional (**Matthew**). Other option is that the ILS polls for the status of a patron's load (by ID) when the patron logs in.	
9	8-Aug	Prevent "select your library" pulldown when possible	Is there some mechanism we could come up with that would make this ugliness unnecessary? Next step- **Xan** to add some language to "recommendation best practices" section about the issue and possible solution Update - language has been added but not evaluated	Closed
10	21-Aug	"Re-fulfillment"	What is the mechanism for an ILS to get a refreshed set of URLs for an item that's checked out? Consider especially a long video that may need re-fulfillment during playback. Consider allowing an item-checkout call to be placed again to get an refreshed URL Added LookupItem to specification	Closed
11	24-Oct	Request ID in CheckOutItem?	It seems that for completeness, the request ID of the request being satisfied by a checkout should be included in this message. TODO: Verify that the field exists in the LCF message (**Matthew**) *A checkout creates a loan id - which can be used to retrieve status of the loan - does this suffice?*	Open

			(CheckOut is really a creation of a loan with its unique URL)	
12	24-Oct	Request ID in SendUserNotice?	If SendUserNotice includes the Request ID, then the ILS should be in a position to know everything about the notification it's supposed to execute. TODO: Verify that the field exists in the LCF message (**Matthew**) *Not currently - I will raise this at the LCF technical panel this Friday (9/11)*	Open
13	24-Oct	Inform ILS about a request	We might need a "wrong-way" message in order for the eContent vendor to inform the ILS about a request that's been created. TODO: Identify the LCF call which will allow the e-content vendor to create/reflect the request in the ILS (**Matthew :)**) *Update Loan Request and Response cover this for checkout requests. For reservation requests a similar approach would work (to be added!)*	Open
14	07-Nov	Best Practices	How do we better reflect the business use case/requirements and best practices in the document? Best practices section needs to be massaged/reworked/restructured to tell the story better TODO: Review after progress is made on #15	Open

15	07-Nov	Structure of the document	The document's structure needs to be reviewed now that we've added lots of content TODO: **Christine** to review structure and recommend changes (Michael and Jane volunteered to review as well.)	Open
16	07-Nov	Inform ILS about request cancellation	See item #13 - we should also have the eContent vendor inform the ILS when a reserve is cancelled.	Open

Text about pseudonymous vs anonymous - to be put somewhere, probably after modifications depending on NCIP/LCF decision:

There are two different patron privacy modes in FASTEN. Both are very good: even in the weaker "pseudonymous mode", no personal patron information is sent. The stronger "anonymous mode" prevents reading history from being connected to individuals at all, at the possible tradeoff of some functionality. The librarian (via the ILS) unilaterally chooses which mode to use: the eContent vendor's behavior is the same in either case.

Pseudonymous mode - The patron identifier which the ILS uses in the API is an identifier generated by the ILS. This identifier should be a random string which is stored in the ILS to map to the patron. Each patron should have a unique identifier for each eContent vendor.

Anonymous mode - A new randomly-generated identifier, as described above, is generated for each interaction. The major downside is that patrons will only be able to see their list of items currently checked out via the ILS. If there is an option for interacting with the eContent vendor directly (via an app, for example), then that app will not be able to show items currently out ix`f anonymous mode is in use.

Differences between NCIP and LCF regarding patron privacy modes:
* With LCF in anonymous mode, there is no need to generate or store patron identifiers. Instead, loan identifiers should be stored and attached to patron records. Both NCIP and LCF support request identifiers, so those request identifiers should be stored by the ILS and attached to patrons in either case.

* NCIP does not have the concept of loan identifiers. The following messages require a patron ID and an item ID, where a loan ID would be preferable:
  - CheckInItem
  - RenewItem
  - LookupItem

In order to facilitate these messages in anonymous mode, the ILS should store the random identifer that was used for the CheckOutItem message and associate it with the patron. Effectively, the combination of the random patron ID and the item ID becomes a loan ID.