

Ex Libris Aleph ILS Support for XC NCIP Toolkit Design and Documentation

Author:

Richard P Johnson

University of Notre Dame

Library Systems Department

213 Hesburgh Library

Notre Dame, IN 46556

Doc Version 0.3

06/25/09

1 of 10

Table of Contents

1	Introduction.....	3
2	Implementation.....	3
2.1	Core Components.....	3
2.1.1	AlephInterface.....	3
2.1.2	AlephMediator.....	3
2.1.3	XService.....	4
2.1.4	Sequence of Events.....	4
2.2	Aleph Objects.....	4
2.2.1	AlephUser.....	4
2.2.2	AlephItem.....	5
2.2.3	AlephAgency.....	5
2.3	Utility Classes.....	6
2.3.1	AlephAgencyFactory.....	6
2.3.2	AlephUserFactory.....	6
2.3.3	AlephItemFactory.....	6
2.3.4	AlephMediatorFactory.....	6
2.3.5	XServiceFactory.....	6
2.3.6	XMLParserUtil.....	6
2.4	Other Classes.....	7
2.4.1	AlephException.....	7
2.4.2	AlephConstants.....	7
3	Build.....	7
4	Configuration.....	7
4.1	Aleph X-Server Configuration.....	7
4.2	log4j.config.txt.....	7
4.3	NCIPToolkit_config.xml.....	8
5	Deploy.....	9
6	Unsupported Operations.....	10

1 Introduction

This document defines an implementation to add Ex Libris Aleph ILS support to the XC NCIP Toolkit. To add any new ILS support in the NCIP Toolkit, it merely requires that you implement an ILS specific version of the ILSInterface within the toolkit. Once available, the NCIP Toolkit can be configured to use this new class to communicate with Aleph.

This implementation relies on Aleph's X-Server for communication with Aleph. The X-Server is a custom web server running within Aleph that receives X-Service calls which then in turn execute an action and/or retrieve information. An X-Service call is initiated by sending an http request to the X-Server and then an appropriate xml response is returned. This mechanism supports nearly all of the functionality of the ILSInterface definition. Any unsupported functionality will be detailed below.

*Please note that this implementation may be leveraged for other uses besides NCIP.

The NCIP Toolkit's ILSInterface requires implementing the following methods (for more information look at NCIPToolkitDeveloperDocumentation.doc):

- authenticateUser
- authenticateUserAgainstExternalLDAP
- lookupUser
- xcLookupUser
- lookupItem
- xcGetAvailability
- renewItem
- requestItem
- cancelRequest
- recallItem
- cancelRecall

2 Implementation

2.1 Core Components

This Aleph specific implementation of the ILSInterface contains the following main components:

2.1.1 AlephInterface

It implements ILSInterface and receives incoming requests from the NCIP toolkit to perform actions against Aleph and/or retrieve information. It then makes calls to the AlephMediator to process the NCIP request. On response from the AlephMediator it will translate Aleph java objects returned to NCIP java objects and then return the NCIP objects to the NCIP toolkit.

2.1.2 AlephMediator

This is an extra layer of abstraction between X-Service calls and the AlephInterface. It will

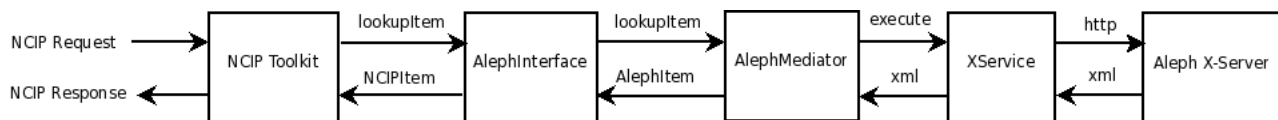
receive requests from the AlephInterface to perform necessary steps for an NCIP request. Therefore the AlephMediator contains corresponding methods to the NCIP methods in the AlephInterface, but it is not always a one-to-one ratio. Based on the method called it will make one or more X-Service calls to the Aleph X-Server.

2.1.3 XService

It is used by the AlephMediator to execute an X-Service call against an Aleph X-Server.

2.1.4 Sequence of Events

An example sequence of events is below (in this case it is lookup item):



2.2 Aleph Objects

Each NCIP method call either returns a user, item or date. For example, lookup user would return a user if found, and lookup item would return an item if found. Because Aleph supports housing multiple institutions in one Aleph instance through the use of sublibraries, each user and item are in the context of the institution. The NCIP toolkit refers to this as an agency so we will use that term as well.

In the NCIP Toolkit users, items, and agencies are represented by NCIPUser, NCIPItem, and NCIPAgency objects. In order to fully abstract the AlephMediator from the NCIP Toolkit, there are corresponding AlephUser, AlephItem, and AlephAgency objects. This is useful because there are subtle differences between how information is stored between Aleph and XC.

These classes are detailed below:

2.2.1 AlephUser

It is returned by authenticateUser and lookupUser methods in the AlephMediator. An AlephUser may have the following values set:

- username (patron id within Aleph)
- full name
- address
- email address
- authenticated username (may be different from username if authenticated via LDAP, etc.)
- agency id (should correspond with an agency defined in the configuration steps mentioned in the Configuration section below)
- balance (balance of any outstanding fines)
- list of any requested items (list of AlephItems)

- list of any loaned items (list of AlephItems)
- list of any fined items (list of AlephItems)
- list of any blocks
- list of any notes

2.2.2 AlephItem

It is returned by methods such as lookupItem and contains the following possible values:

- itemId (equivalent to adm id + sequence number)
- bibId
- holdingsId
- description
- location
- call number
- author
- isbn
- medium type (ie. book, periodical, etc.)
- docNumber (adm id)
- seqNumber (sequence number)
- publisher
- series
- title
- circulationStatus (will be set to "A" if checked out)
- electronicResource (a path to an online resource if any)
- barcode
- sessionId (X-Service session id from original response)
- dateHoldRequested (populated if a requested item)
- dateAvailablePickup (may be populated if a requested item)
- holdRequestId (id for the hold request record)
- dueDate (date due if on loan)
- fineAmount (set if this item is a fined item)
- fineAccrualDate (date fine incurred)
- fineStatus (indicates whether a fine is paid, unpaid, etc.)
- creditDebit (if value of "D" then it is a debit fine, if "C" it is a credit to the patron account)
- holdQueueLength (currently unsupported)
- List of Borrowing Users (list of AlephUsers...currently unsupported)
- List of Requesting User (list of AlephUsers...currently unsupported)
- availability (set to Available, Possibly Available, Unavailable, or Unknown)

2.2.3 AlephAgency

Maps an agency id to an Adm Library, Bib Library, and Holding Library in Aleph. The list of Aleph agencies is loaded from the NCIP Toolkit configuration explained in the Configuration section below. Whenever a request comes into the AlephInterface for a specific agency, an AlephAgency object is used to determine which adm library, bib library, and/or holding library

should be passed to the X-Server in the X-Service call. The AlephAgency object contains the following properties:

- agencyId
- admLibrary
- bibLibrary
- holdingsLibrary

2.3 Utility Classes

There are also several classes that facilitate object creation and parsing XML responses from X-Service calls:

2.3.1 AlephAgencyFactory

Creates an AlephAgency based on the agency id passed in. Each agency is then mapped to an Adm Library, Bib Library, and Holding Library in Aleph. This mapping is determined based on configuration within the NCIP toolkit. There is more detail on this in the NCIP toolkit.

2.3.2 AlephUserFactory

Will take an XML response from either a bor-auth (authenticate) or bor-info (lookup user) X-Service call, parse through the xml data and create an AlephUser object with available data populated. The data may include loaned items, fines, hold requests, etc.

2.3.3 AlephItemFactory

Will take an XML response from multiple types of X-Service calls and return either an AlephItem or a list of AlephItems. For example, AlephItemFactory is called by AlephUserFactory when there are loaned items linked to a user.

2.3.4 AlephMediatorFactory

Returns either a new or existing instance of an AlephMediator. It is currently only used by the AlephInterface.

2.3.5 XServiceFactory

It will create an XService object to make a specific X-Service call. There is a separate method for creating each type of X-Service call supported. The parameters will contain all necessary information for that X-Service call, except for X-Server name and port (these are passed to XService object at execute time).

2.3.6 XMLParserUtil

Common parsing operations across the AlephUserFactory and AlephItemFactory are

concentrated into the XMLParserUtil class for simplicity.

2.4 Other Classes

2.4.1 AlephException

An exception thrown by any XML parsing methods that contains an error returned in an XML response from an X-Service call to Aleph.

2.4.2 AlephConstants

Any Aleph specific static constant values are defined here, such as ones used in XML parsing, etc.

3 Build

All Aleph specific class files (including AlephInterface) will be compiled into the standard NCIPToolkit war file.

4 Configuration

Once you download the NCIPToolkit.war file you will need to modify some configuration files before deploying it. You must first unwar the contents of the file. You will then modify two files (details follow), and then rewar the contents. This newly configured war file can then be deployed to any JEE server such as Tomcat, Websphere, WebLogic, etc. to run the NCIPToolkit.

4.1 Aleph X-Server Configuration

The Aleph NCIP driver communicates with Aleph via the X-Server. Therefore, you must have an instance of the Aleph X-Server running for the NCIPToolkit and Aleph integration to work. The X-Server will only accept X-Service calls from trusted IP addresses. Therefore, the IP address for the server that will run the NCIP Toolkit needs to be added to the allowable IP addresses table on your Aleph X-Server machine. Please refer to Ex Libris Aleph documentation for how to do this.

4.2 log4j.config.txt

The NCIPToolkit leverages a Java logging framework from Apache called log4j. This can be configured by modifying the WEB-INF/classes/log4j.config.txt. At minimum you will want to modify the log4j.appender.file.File to the path of your intended log file location on your web server that will run the NCIPToolkit. log4j also offers six different levels of logging:

- [TRACE](#)
- [DEBUG](#)
- [INFO](#)
- [WARN](#)

- [ERROR](#)
- [FATAL](#)

Trace or debug will produce the most log messages and are typically used during development. Most likely in production you will want to use a level of INFO for your logging. You can also use WARN or ERROR if INFO produces more logging than you would like. The logging level can be set by modifying the log4j.rootLogger property. Here is an example entry:

```
log4j.rootLogger=DEBUG, file
```

4.3 NCIPToolkit_config.xml

Within the NCIP Toolkit WEB-INF/classes/NCIPToolkit_config.xml configuration file there are some Aleph specific properties that need to be configured. At minimum you will need to correctly configure the properties listed below (Please refer to standard NCIP documentation for details on other properties in this file):

- NCIPToolkitUrl

: should be set to the server URL for your NCIP instance such as

<http://myserver.domain:8080/NCIPToolkit/ncipToolkit>

- SSLNCIPToolkitUrl

: Similar to NCIPToolkitUrl property except using SSL such as

<https://myserver.domain:8443/NCIPToolkit/ncipToolkit>

- NCIPSchemeFolderUrl

: URL to NCIP scheme that will be used (does not have to be from same NCIP instance).

Example: <http://myserver.domain:8080/ncip/schemes>

- LoggerConfigFileLocation: Should be set to absolute path of the log4j.config.txt file where the NCIPToolkit will be deployed. This path is most likely the deploy location plus WEB-INF/classes/log4j.config.txt. Example:
/usr/share/tomcat6/webapps/NCIPToolkit/WEB-INF/classes/log4j.config.txt
- ILSType: should be set to "Aleph"
- ILSDefaultAgency: should be set to the default agency and match an agency name set in the AlephILSAgency property below.
- AlephXServerName: the name of your Aleph X-Server
- AlephXServerPort: the port for your Aleph X-Server (usually set to 8991)
- AlephCurrencyCode: the currency code for Aleph (example USD for US Dollars)
- AlephILSAgency: The NCIP driver supports a single Aleph instance with multiple institutions. Therefore, all institutions should in this property as a comma-separated list
- AlephAdmLibrary: a comma-separated list of Aleph Adm libraries for the agencies defined in AlephILSAgency. The corresponding Adm library to agency should appear in this list in the same order as the agency in AlephILSAgency.

- AlephBibLibrary: a comma-separated list of Aleph Bib libraries for the agencies defined in AlephILSAgency. The corresponding Bib library to agency should appear in this list in the same order as the agency in AlephILSAgency.
- AlephHoldLibrary: a comma-separated list of Aleph Holding libraries for the agencies defined in AlephILSAgency. The corresponding Holding library to agency should appear in this list in the same order as the agency in AlephILSAgency.

Sample configuration:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<properties>
  <variables>
  </variables>
  <category name="general">
    ...

    <property name="ILSType" value="Aleph" />
    <property name="ILSDefaultAgency" value="University of Notre Dame" />

    <property name="AlephXServerName" value="barnabas.library.nd.edu" />
    <property name="AlephXServerPort" value="8991" />

    <property name="AlephCurrencyCode" value="USD" />

    <!--To add an additional agency, just separate values by "," below-->

    <property name="AlephILSAgency" value="University of Notre Dame,Saint Mary's
College,Holy Cross College,Bethel College" />
    <property name="AlephAdmLibrary" value="NDU50,SMC50,HCC50,BCI50" />
    <property name="AlephBibLibrary" value="NDU01,SMC01,HCC01,BCI01" />
    <property name="AlephHoldLibrary" value="NDU60,SMC60,HCC60,BCI60" />
  </category>
</properties>
```

5 Deploy

Once the WEB-INF/classes/NCIPToolkit_config.xml and the WEB-INF/classes/log4j.config.txt are properly configured, the NCIPToolkit can be wrapped in a war file containing the changes and deployed to any JEE server such as Tomcat, Websphere, WebLogic, etc. (Note: the deployed server's IP address should be one configured as allowable by the X-Server in the configuration steps above. If not, you will need to add it as well to the allowable IP addresses).

Please refer to the NCIP Toolkit documentation any more deployment information.

6 Unsupported Operations

There are some ILSInterface that are completely unsupported, and for other methods certain data or types of operations are not supported. These methods use operations that are currently unavailable through X-Services in Aleph. For any methods completely unsupported, AlephInterface throws an unsupported exception. If it is only partial support, then any data that can be provided will be.

- Completely unsupported methods:
 - recallItem
 - cancelRecallItem
 - acceptItem
 - checkInItem
 - checkOutItem
 - createItem
 - xcOpenUrlRenewItem
 - xcOpenUrlRequestItem
- Partial support
 - lookupItem (partial support except items below)
 - Get hold queue length
 - Get borrowing users
 - Get requesting Users
 - xcLookupUser (partial support except items below)
 - Get Recalled Items
 - requestItem (partial support except items below)
 - Call Slip Request
 - Set Pickup Location
 - Set Pickup Expiration Date
 - Set Comments
 - cancelRequestItem (partial support except items below)
 - Cancelling by request is not supported (by item id and bib id only are supported)