# Aleph API Adapter User’s Guide

Version 1.6, December 2014

Contents

[Overview 2](#_Toc389476134)

[Design 3](#_Toc389476135)

[Environment 4](#_Toc389476136)

[Requirements 5](#_Toc389476137)

[Installation 6](#_Toc389476138)

[Options 7](#_Toc389476139)

# Overview

The Aleph API Adapter is a middleware script positioned between the Aleph REST API and any external system that calls the REST API for services. It has a small footprint, is easy to install, and is unobtrusive in operation. Use of the adapter confers these benefits on Aleph customers.

1. Insulates an external system from a competing vendor’s product and provides an avenue for problem circumvention that does not depend on either Aleph or the vendor.
2. Can minimize the latency of web traffic.  Some queries require several separate operations against the Aleph API. If the external system’s code runs directly against the Aleph API, every one of those queries must make a round trip between the external server and the Aleph server.  With the intermediate layer, one call from the external server to the adapter can result in several calls to the Aleph API, all of which are done on the Aleph server using ‘localhost’.  None of the ‘localhost’ API calls goes out on the wire and consequently are very fast.
3. APIs change over time and across releases, contain bugs, and sometimes are incomplete.  An intermediate layer can mask changes unless or until they prove useful, and can compensate for missing functionality.
4. Simplifies access control.  Aleph controls access to APIs via IP addresses.  Since the X-server is required in order to get Aleph ids for use with the REST API, IP addresses for external servers must be maintained in two places within Aleph: in $alephe\_tab/server\_ip\_allowed and in the Tomcat/JBoss configuration. It is easier to maintain whitelists of approved IP addresses in one place, the adapter, than it is in Aleph.  It also gives the adapter more flexibility in serving other processes.
5. Provides more flexibility for the external system.  If there is a need for Aleph data to be filtered, or augmented from other local systems, the adapter is a perfect place to do this without requiring special programming on the external server.

# Design

The adapter was built to achieve these design goals.

1. Easy to integrate into the Aleph server environment.
2. Have a small footprint and be unobtrusive in its operation.
3. Return results identical to those generated by the Aleph REST API unless intentionally altered by the customer.
4. Does not interfere in any way with the standard use of the Aleph REST API through the Tomcat/JBOSS server.
5. Consumes Aleph REST API URL syntax without modification.
6. Function in a transparent pass-through mode for all REST API services except for those intentionally modified by the host site.
7. Can run on ports 80, 443, 8991, or any other port that may be in use for the Aleph OPAC.

# Environment

The adapter is intended to run from the cgi-bin subdirectory on Aleph’s Apache server. During installation, it is activated by the completion of two steps.

1. Some Apache configuration directives must be activated in Apache to trap REST URLs and route them to the adapter (see the Installation section). If the .htaccess file is in use in the document root of Aleph’s Apache server they can be included there. Otherwise they will need to be included in the main Apache configuration file under a stanza for the document root directory.
2. The adapter will run on SSL (recommended), port 80, or other ports. For example, some Aleph sites may use port 8991 for their OPAC instead of port 80. The adapter should run on all such configurations without modification.  
     
   Examples:  
     
   This URL calls the REST API directly using port 1891:  
   **http://<server name>:1891/rest-dlf/patron/<Aleph id>/patronInformation**  
   This URL calls the adapter using SSL:  
   **http**s**://<server name>/rest-dlf/patron/<Aleph id>/patronInformation**This URL calls the adapter without SSL using port 80:  
   **http://<server name>/rest-dlf/patron/<Aleph id>/patronInformation**This URL calls the adapter without SSL using port 8991:  
   **http://<server name>:8991/rest-dlf/patron/<Aleph id>/patronInformation**

When the adapter is implemented it does not foreclose the direct use of the REST API on its specified port. The adapter does not interfere with Tomcat/JBOSS operation in any way.

# Requirements

1. The adapter is written in Perl. Most Aleph servers have two instances of Perl installed: one that is installed with the OS and one that is installed as part of Aleph. The adapter defaults to running under the standard Perl instance installed with the OS. It is not dependent on the Aleph Perl instance. If the adapter is intended to run under the Aleph-specific Perl installation, the first lines of **api\_adapter.template, install\_adapter.pl**, and **validate.pl** must be altered before the installation scripts are run.  
     
   The adapter requires that these four packages be present in the Perl instance where it will run.   
   **HTTP::Request  
   LWP::UserAgent  
   Switch  
   POSIX**If the X-server is not licensed on the Aleph server, an SQL lookup will be implemented. In this case, these two additional Perl packages will be required:  
   **DBI  
   DBD::Oracle**
2. JBOSS configuration  
   The adapter uses the server name ‘localhost’ when addressing Aleph’s REST API. Consequently, IP address **127.0.0.1** must be included in Aleph’s JBOSS configuration in /exlibris/aleph/a<nn>\_<n>/ng/aleph/home/system/thirdparty/openserver/server/default/deploy/jbossweb.sar/server.xml. See Ex Libris manual How to Configure the JBoss Server in Aleph.
3. Access to the X-server bor-by-key function is required for conversion of aliases into aleph\_ids. Column 5 of tab\_bor\_id.<lang> in the ADM library will need to be set to ‘Y’ for each type of identifier that will be submitted from the remote server.   
     
   If the adapter is going to be run with id\_translation turned off, the X-server whitelist in $alephe\_tab/server\_ip\_allowed must contain IP addresses for all remote servers that will need to look up aleph\_ids.

# Installation

1. Acquire the files from GitHub.
2. Log into the Aleph system as user ‘aleph’.
3. Change directory to the cgi-bin subdirectory of Aleph’s Apache server, and copy the files to that directory.
4. Add these directives to the .htaccess file in Apache’s DocumentRoot directory e.g. **/exlibris/aleph/u22\_1/alephe/apache/htdocs/.htaccess.**

**# Rewrite rules for Aleph API Adapter  
RewriteEngine On**

**RewriteBase /  
RewriteCond %{REQUEST\_URI} ^/rest-dlf/  
RewriteRule (rest-dlf/.\*) /cgi-bin/api\_adapter.cgi?parm1=$1 [E,L]**If the ‘**RewriteEngine’ and ‘RewriteBase’** directives already exist in .htaccess, they do not need to be repeated.  
  
If the use of htdocs/.htaccess is not enabled on the Aleph system in question, the rewrite rules can either be included at the appropriate place in the Apache configuration, or the use of .htaccess enabled by making a change in Apache’s httpd.conf file. The default version of httpd.conf contains this stanza:

<Directory "/exlibris/aleph/u22\_1/alephe/apache/htdocs">

Options FollowSymLinks MultiViews

**AllowOverride None**

Order Deny,Allow

Allow from all

</Directory>

Change ‘AllowOverride None’ to ‘AllowOverride All’. Apache should be restarted after making this change.

1. Run the installation script with this command:  
   **./install\_adapter.pl**   
    The installation script will perform the following operations:
   1. Get the PWD environment variable and extract the version token from the path.
   2. Prompt the installer for the institution’s name. This will be used in the adapter’s response to an ‘ilsinstance’ call, the one non-Aleph call that it accepts. Here is an example of the structure returned by the ‘ilsinstance’ call, reporting the institution’s name, the version of Aleph, and a few site-specific Aleph variables:  
      <version>  
       <ilsInstitutionName>MIT</ilsInstitutionName>  
       <ilsVersion>22.0.3</ilsVersion>  
       <locale>en\_US</locale>  
       <timeZone>Eastern Standard Time</timeZone>  
       <timeZoneCode>EST</timeZoneCode>  
       <timeZoneGMT>-5</timeZoneGMT>  
       <currencyCode>USD</currencyCode>  
      </version>
   3. Prompt the installer for the Tomcat/JBOSS port number.
   4. Prompt the installer for the status of the X-server. If the X-server is not licensed for use on the Aleph server, an alternative method of converting patron identifiers into Aleph ids via SQL must be configured.
   5. If the X-server is not available, three other pieces of information must be acquired.
      1. Prompt the installer for the Oracle user id and password of the ADM library, which is usually <xxx>50.
      2. Prompt for the z308 prefix of the identifiers that the external system will submit for translation. This refers to the first two characters in each patron record in the z308 Oracle table. For example, ‘01’, or ’02’, or ‘03’ etc.
   6. Prompt the installer for an Aleph patron id to be used in verifying the installation.
   7. Prompt the installer for the IP address of each remote server that will call the adapter.
   8. Generate the ‘**get\_aleph\_info.csh**’ shell script from a template, interpolating the version token as needed.
   9. Generate the ‘**api\_adapter.cgi**’ script from a template, interpolating the server IP addresses into the whitelist and the JBOSS port number into the appropriate variable.
   10. If the X-server is not available, generate the SQL lookup scripts.
       1. Generate the ‘**sql\_lookup.csh**’ script from a template, interpolating the Aleph version into file paths.
       2. Generate the ‘**sql\_lookup.cgi**’ script from a template, interpolating the Oracle user id, the Oracle password, and the z308 prefix into the appropriate variables.
   11. Run **validate.pl** to test the installation. Validate.pl will perform the following operations:
       1. Verify the presence of the required Perl packages and report success or failure.
       2. Using the Aleph id supplied by the installer, construct and submit a URL to the X-server and report success or failure.
       3. Using the Aleph id supplied by the installer, construct and submit a URL to the REST API and report success or failure.
       4. Verify the operation of the get\_aleph\_info.csh shell script.
2. If no errors are reported by the installation and validation scripts, the installation is complete. The validation script generates and displays a test URL. Test this URL from any server whose IP address was specified during installation. It can be tested from a browser if the browser’s IP address is part of the server whitelist.

# Options

Id\_translation

This option is useful when the external system submits REST URLs that contain an alternate identifier instead of an Aleph id. With id\_translation set on (the default), the adapter extracts the incoming alternate identifier from the URL, calls the X-server to get the corresponding Aleph id, and substitutes the Aleph id for the alternate identifier in the call to the REST API. If this conversion of identifiers is not needed, id\_translation can be turned off.

If a remote server needs to call the REST API with an alternate identifier, one of four scenarios must be followed.

1. The adapter is not installed.  
   In this case, the remote server must first call the X-server’s ‘bor-by-key’ function, passing the alternate identifier and receiving the corresponding Aleph id. The Aleph id can then be interpolated into subsequent calls to the REST API.
2. The adapter is installed, but id\_translation is turned off.  
   This case requires the same process as case #1.
3. The adapter is installed and id\_translation is turned on (default)  
   In this case the remote server calls the adapter with a REST URL containing an alternate identifier. The adapter extracts the alternate identifier, calls the X-server to retrieve the corresponding Aleph id, replaces the alternate identifier in the URL with the Aleph id, and sends the enhanced URL on to the REST API. Since the X-server call in this case is done on the local server without going out on the network it is very fast.
4. The adapter is installed and id\_translation is turned on (default), but the X-server is not licensed for use on the Aleph server.  
   This case is the same as as case #3 except that the adapter calls SQL lookup scripts instead of the X-server for id translation.

Under what circumstances should id\_translation be turned off? If the remote server is populating all its REST URLs with Aleph ids, then the translation step is wasted processing time. In that case, id\_translation can be turned off with these actions:

1. Make a backup copy of api\_adapter.cgi.
2. Edit api\_adapter.cgi
3. Find this text ‘my $id\_translation = 1;’ and change the 1 to a zero.
4. Save the file.