

Vendors and Products

Abstract

This chapter of “Opening Up Library Systems through Web Services and SOA: Hype or Reality” takes an in-depth look at what is currently available on the ILS market, what services are offered in terms of APIs and customizability, and how users have reacted to the products and services. Vendors covered in this chapter include Ex Libris, The Library Corporation, Innovative Interfaces, Polaris, SirsiDynix, Talis, and VTLS.

In this section we will take a look at some of the major integrated library systems and explore the extent to which they offer APIs. Each vendor was contacted with a request to respond to a set of questions that elicit information about the APIs offered by the systems that they support. The responses received from each product representative are presented as received.

Ex Libris

Ex Libris, based in Israel, specializes in providing automation software to research libraries and consortia, with a customer base that includes some of the largest libraries in the world. The libraries that use this company’s products fall well within the profile of those that expect to extend or adapt the software. These libraries tend to have very complex automation requirements that may not be met by any off-the-shelf solution. By the nature of the

Ex Libris
www.exlibris.co.il

demographics of its customer base, Ex Libris faces a very high demand for extensible systems.

Ex Libris offers a variety of automation products. The company developed the Aleph ILS in the early 1980s and has advanced this product through several generations of technologies. Aleph has been deployed throughout the world, primarily by academic and national libraries, but also by consortia that include public libraries. Voyager, developed by Chicago-based Endeavor Information Systems, was acquired by Ex Libris in 2006. Voyager finds use primarily in academic libraries in the United States, the United Kingdom, and Australia; it is the automation system used by the Library of Congress. Other products offered by the company include the SFX OpenURL link server; MetaLib, a federated search product; Verde for electronic resource management; the Primo discovery interface; and most recently, Rosetta, a digital preservation system.

Given Ex Libris’s clientele, the company faces the challenge of offering not only products with sophisticated functionality, but ones that can be extended to handle complex and specialized library automation scenarios. One of the primary approaches that the company has taken to satisfy these requirements has been an evolving set of APIs.

Ex Libris offers a diverse set of products, created over a long time frame that spans many different cycles of technology architectures. Aleph, for example, traces its roots to the early 1980s as a mainframe product written in COBOL. This long history has allowed it to steadily accumulate new functionality, growing into one of the most sophisticated and feature-rich ILS products on the market, capable of serving the world’s largest and most complex libraries. While it has been completely reworked over its long history, vestiges of technologies from past

ages remain. Voyager, created in the mid-1990s as a client/server automation system, embodies a significantly different architecture and programming style. SFX, the company's OpenURL link server, was originally created at the University of Ghent in Belgium. Though it has been largely rewritten, its codebase differs substantially from the company's other products. MetaLib, the company's federated search platform, and DigiTool, a digital asset management system, though developed somewhat more recently, still predate current service-oriented software.

Throughout its corporate history, Ex Libris has taken at least some measures to give its customer libraries the ability to work with its products beyond the user interfaces delivered with the system. Each of these products has offered a set of APIs appropriate for its technical heritage. While each product offers APIs in some way and to varying extents, when APIs are considered across the individual products, they are disjointed and inconsistent. While libraries might appreciate that each application offers some flavor of APIs, they may be less than delighted at the complexities of programming against each product in a different way.

In the last year, Ex Libris has articulated a strategic direction of technology based on a more thorough and consistent delivery of APIs. This strategy involves exposing a more comprehensive array of functionality in its products through APIs and delivering them more consistently. As the company develops new versions of its current products as well as entirely new products, it will work toward creating a more uniform and coherent set of APIs, delivered through a consistent and modern Web services approach.

In June 2008, Ex Libris announced its "open-platform program," which stated that its corporate strategy was to develop open interfaces to its products, to provide full documentation for its APIs, and to evolve the APIs across all its products to a more consistent and unified structure. The program specifies that forward development will emphasize a service-oriented architecture. Oren Beit-Arie, Ex Libris chief strategy officer, has been one of the guiding forces behind the company's support of open APIs; Tamar Sedeh, director of marketing, has been given responsibility for the open-platform program.¹

As part of the fulfillment of its open-platform program, Ex Libris created a workspace called "EL Commons" as the focal point for programmers involved with Ex Libris products. EL Commons includes a repository to gather together and make available any programs or scripts that make use of the APIs associated with any of the Ex Libris suite of products, documentation for the APIs associated with each of the products, and forums to facilitate enquiries and discussion surrounding their use. At the of this report's publication, Ex Libris restricts access to EL Commons to its direct customers, but intends to open its access to noncustomers in the near future to accommodate those involved

with other organizations that want to develop interoperable applications with Ex Libris products.²

Survey Responses³

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

Ex Libris offers open interfaces for all of the company products, enabling libraries to customize our products, extend them by adding locally developed or third party functionality, and integrate entire products or parts of them with other systems in the library. We divide the open interfaces into several categories:

Application Programming Interface (API). Web services (wrapped in SOAP), X-services, and RESTful services.

The first APIs that we developed a decade ago for the Aleph integrated library system, X-Services, were http/XML API. X-services are available for most other Ex Libris products as well, and serve an important role in initiatives such as David Walker's Xerxes system, developed as an open source system using metasearch functionality through MetaLib X-services. The next generation of APIs was developed as Web services wrapped in SOAP, and today new APIs are developed as representational state transfer (REST) services. For example, the DLF-DI services are now being implemented as RESTful services. All APIs (X-Services, Web services wrapped in SOAP, and RESTful services) are available on the EL Commons collaborative Web site, which will be described later in this document, and enable customers to use them in whatever programming language they wish to use.

Plug-ins. Routines provided by Ex Libris that interact with our applications in a predefined way determined by our applications, to provide a specific function. Institutions or a 3rd party can modify our plug-ins to create new capabilities. For example, by modifying the out-of-the-box "enrich indexing" plug-in, Primo customers can enrich the records that they harvest with information such as reviews, book covers, and tables of contents from the external resources that are relevant to them.

Adapters. Routines developed by Ex Libris to bridge between our applications and 3rd party applications and allow applications to interoperate even when having incompatible interfaces. For example, the Deep Search adapter enables Primo to search in a remote resource using the API of

that resource. Another example would be a MetaLib external program that enables MetaLib to send a search request to an information resource that does not support standard protocols and obtain the results returned by that resource. Customers can modify our adapters and develop their adapters as well, based on our adapters.

Deep Links. URLs that point to a specific HTML page, such as result list or the full display of an item.

Reporting Schema. A set of Oracle views that provides read-only access to the data, for reporting purposes.

We provide comprehensive documentation, including examples, for all interfaces. In some cases we provide additional programming tools such as utilities and libraries of functions or classes that together form a software development kit (SDK).

*What is the business model for your APIs?
Is access to the APIs included with the base product, or is it an extra-cost option?*

Today all open interfaces are offered as part of the core product and we do not charge any license fees for them.

*Do you publish documentation for your APIs?
Is it available only to licensed customers, or to third-party developers as well?*

Comprehensive and consistent documentation of the open interfaces across all products is available on a collaborative Web site—EL Commons—that serves as a communication channel among Ex Libris community members (library developers and Ex Libris developers). The EL Commons Web site consists of two parts: a wiki, defined and administrated by the customer user groups (IGeLU and ELUNA), and the Developer Zone. The documentation of the open interfaces is available on the Developer Zone, along with contributions posted by customers and Ex Libris. By default, contributions are open source and are assigned BSD license; however, contributors can change the license as required.

The EL Commons Web site is available to all Ex Libris customers, regardless of the product they deploy. Parts of the site will be open to all by October 2009. We enable access to third party developers if requested by our customers.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

As explained above, Ex Libris provides APIs of three types, all transferring data over the Web. X-Services are based on simple http/XML data transfer, whereas the other APIs are either wrapped in SOAP or adhere to the REST network architecture.

Can the API create and modify data, or is it read-only?

The open interfaces enable the full spectrum of functionality, including the creation and the modification of data.

Describe any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

There are many examples on the EL Commons Developer Zone, where customers post their code and enable it to other customers. Most code contributions are developed to extend the Ex Libris solutions by adding functionality. A few recent examples:

- OPAC spelling suggestions using Yahoo! Web service *JSON* (Aleph; Mark Watmough, Napier University, UK)
- ticTOCs and embedded SFX services in OPAC (Aleph; Daniel Forsman, Jonkoping University, Sweden)
- WebVoyage 7 (Tomcat) Addition of OCLC “Cite This Item” Link (Voyager; Laura Guy, Colorado School of Mines, USA)
- Patron History Browser (Voyager; Rhoel Gerona, Bay of Plenty Polytechnic, New Zealand)
- iPhone view (Primo; contributed by Ex Libris)
- Add Primo tags from an external source (Primo; John Osborn, University of Iowa, USA)
- doSFXTransfer (SFX; Charlie Ambrose, Aarlin, Australia)
- OCLC’s OpenURL Resolver Registry as SFX Target (SFX; Inga Overkamp, Max Planck Society, Germany)
- MetaLib Automatic Monthly Statistics Gathering Script (MetaLib; Ere Majjala, National Library of Finland)
- EZproxy Authentication Adapter for PDS (MetaLib; Ere Majjala, National Library of Finland)

Some projects are on a larger scale. Most notorious examples:

Xerxes—a metasearch system developed by David Walker of Cal State and Jonathan Rochkind of Johns Hopkins University, using X-Services to offer the full spectrum of MetaLib functionality.

Livetrix, (also known as PurpleSearch) another metasearch system, developed at the university of Groningen by Bart Alewijnse and André Keyzer, relying on MetaLib X-Services in a way similar to that of Xerxes

Umlaut, a link resolver front end, developed by Ross Singer while being at Georgia Tech and Jonathan Rochkind from Johns Hopkins University, relying on the SFX APIs

Xerxes
<http://xerxes.calstate.edu>

Livetrix
<http://purplesearch.ub.rug.nl/help.docs>

Umlaut
http://wiki.code4lib.org/index.php/About_Umlaut

In what way do you consider your system one that embraces the service-oriented architecture?

Our newer systems adhere to service-oriented architecture principles, and as we move forward we are putting more emphasis on modularity and on developing our solutions as components that interact through open interfaces. A good example is the way we implemented the inclusion of OPAC functionality in Primo version 3.0 to be released at the end of 2009: we developed a suite of RESTful services for Aleph and for Voyager, and applied the functionality in Primo relying on the interfaces included in these suites.

The development of open interfaces is defined within our design and coding practices.

Are there any other issues regarding APIs that you would like us to be aware of as we develop this report? Are there other approaches that your company supports instead or in addition to APIs for providing end-user access to system data and functionality?

First, I'd like to stress that we at Ex Libris attribute much importance to the openness of our systems. We realize that despite the rich functionality and the flexibility of our solutions, we cannot satisfy all needs of all customers cost-effectively and in a time frame that fits all. We believed in open systems years ago, but in the last 15 months we went through an extensive process in the company, examining our practices and setting in place a framework for defining, developing, and supporting open interfaces. Furthermore, rather than developing these interfaces only so that customers can extend our systems, we rely on them when we develop our systems and we make sure to develop functionality that is relevant to more than one system as a separate module, interoperable through open interfaces. Our open interfaces cover rich functionality that is not limited to end-user access to the system.

In May 2008 I was appointed to lead the open-platform program and now have an overall responsibility for the program within the company. To carry out the program we went through organizational alignment, nominating one of the senior development staff members to head the open-platform program from the development side, and a focal point in every development team. We built the Developer Zone of the EL Commons Web site and we rearranged our documentation of open interfaces so that

all of it is available in one place and in a consistent format. We regularly communicate with customer developers and conduct meetings during user group meetings. In addition, we hold two annual Developer Meets Developers meetings. Up to now we had held two such meetings—one at the Company's headquarters in Jerusalem in November 2008 and the other at the Company's North American R&D center in Chicago in March 2009. The meetings enabled customer developers and Ex Libris developers to communicate and discuss technical and other matters. We believe that the active community of developers is a key component in the open-platform program and we are inspired and impressed by their ideas.

Another thing that I'd like to point out is that the open-platform program enables libraries to enjoy the benefits of commercial products—stability, regular enhancements and upgrades, professional support, rich functionality from the outset, and a clear, committed roadmap—as well as the flexibility and agility of open source. Libraries do not have to commit to one way or another, and can start their innovation at the point that suits them best. For most libraries, engaging in big development processes is not a desired task for several reasons. For example, many libraries do not employ professional software engineers and hence lack the human resources to commit to long and complex development cycles that, in some cases, may not yield the expected results in the time frame set. Therefore, libraries prefer to invest their effort in adapting existing systems to their needs by sophisticated customization and addition of functionality that they develop or acquire. Our openness and tools enable such a process in the best possible way.

And last but not least, the open-platform program enables all libraries, regardless of their size, budget or IT resources to benefit from the creative work done by all developers. That is, libraries that are more limited in their capabilities to develop code can use code written at other institutions or collaborate with others to carry out projects they could not perform themselves.

CASE STUDIES AND CUSTOMER RESPONSES

Duke University Libraries⁴

Ken Mitchell, IT analyst at Duke University Libraries, provided the following response to the questions regarding the APIs offered by Ex Libris:

Just as an overview of how we're using vendor APIs for our library systems, we are currently using a catalog interface that primarily uses Endeca as the search-and-retrieval system, with various hooks into ALEPH for services that are not provided by the Endeca

application (e.g., live circulation statuses, hold-request functionality, course-reserves, etc.). Where possible, we are using web-service/API services for both Endeca and ALEPH (we had to develop our own XML-based web-services layer for Endeca, as the version we are using only provides API services for Java, ASP, and ASP.NET).

We are also using Ex Libris APIs for our MetaLib and SFX applications, and have developed a full interface replacement for MetaLib using those APIs (similar to David Walker's Xerxes project, but with a focus on developing an "integrated" application interface for our catalog/article/databases/etc. services).

Do you feel like you can pretty much do anything you want with the system, or do you feel constrained?

We are definitely constrained in the ALEPH system. The "X-Services" API layer provides access to *some* important functions, but it also:

- (a) Leaves out a wide range of functions (e.g. no "browse"/authority search features, no ability to determine an item's requestability, etc.);
- (b) Has crippled functionality for many of its services (e.g. a hold-request service that does not allow for specifying a pickup location; patron-information services that do not provide sufficient information for identifying local patron credentials, etc.);
- (c) Does not appear to have been road-tested in any meaningful way (e.g. no item-status calls that can provide the appropriate range of item information in a single call; mysteriously hard-coded values in API calls that do not appear in OPAC or GUI routines, etc.)

Are the APIs offered able to address all the data and functionality within the ILS?

No. Not even close. The ALEPH "X-Services" API does not provide the range of functionality required to replace the application's OPAC. Some missing features: hold-request processes (both on the query level [to determine requestability] and on the request level), browse/authority list search-and-retrieval processes; complete item-level information in a single call, etc.

On the flip side, do you feel like your ILS is too closed?

No. ALEPH *does* have an API library (as quirky as it is), and many schools have opted to perform direct database calls to gain access to data and/or functionality that is not accessible through the APIs.

Do you find the APIs offered by the developer of the ILS to be well documented?

Yes. There are occasional typos, and some undocumented bugs/features for the APIs, but overall ALEPH has a reasonably well-documented set of APIs.

What programming languages or other tools were you able to use to take advantage of these APIs?

We have developed both automated scripts and interface applications taking advantage of the APIs, primarily using Perl, PHP, and Python (often layered under the Django framework), and often using XSL for the presentation components of these applications.

What level of programming proficiency is required: systems librarian with scripting languages, software development engineer, or something in between?

Depending on the purpose of the application, either level of expertise is adequate, since the APIs themselves are accessible via simple URL calls. In general, though, in our environment we are using the skills of software development engineers to develop most of our scripts and/or applications.

What's on your wish list? What kind of APIs would you like to see incorporated into your current or next ILS?

As we look more towards abstracting out the discovery layer and process from our ILS, the primary features we would like to see in our ILS are:

- (a) on-demand harvesting mechanisms (OAI-PMH services; MARC-record delivery systems)
- (b) full-featured holdings/circulation data services (circulation statuses on title and item-levels; hold-request functionality [query and request services])
- (c) specialized/personalized data services that fall outside of routine discovery services (e.g. library patron account information; course reserves data access)


Broome Community College

Mike Curtis, systems librarian at Broome Community College, one of the State University of New York libraries that uses Aleph 500, responded to inquiry regarding ILS APIs, reporting on his experience creating an entirely new OPAC interface using the APIs available with Aleph.

I also used it create RSS feeds for search results. That is demonstrated on the search results page.

The API has a lot of functions for doing circulation transactions, but I haven't used them. Basically, the staff client works just fine for circulation/cataloging procedures. When it doesn't it usually seems that SQL is the answer, or that the system suffers from poor configuration.

Broome Community College
www.sunybroome.edu/library/x/ccl.php

BCC Library Search


Search

• Switch to [Advanced](#) search

New Books











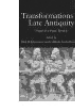















										
The world : a history / Felipe Fern...	Criminal justice : 2005 update / Ja...	New directions for institutional re...	Clinical laboratory science educati...	Access to oral health care / guest ...	Bobby Fischer : profile of a prodig...	101 chess opening traps / Steve Gid...	Science and civilization in China.	Pawn power in chess / by Hans Kmoch...	The theory of the gene.	Transformation of late antiquity :...
										
Treasure chest of six sigma growth ...	Henry James; : an introduction and ...	The psychology of eating and drinkl...	Henry James; : an introduction and ...	Readings in political philosophy, /...	CSG state directory : Directory III...	Containing costs and improving prod...	The Book of the states 2008 / [comp...	Stripping Gypsy : the life of Gypsy...	Digging : the Afro-American soul of...	The encyclopedia of New York City /...
										
Developing effective policy analysi...	Making music with the young child w...	Excel 2007 / Matthew MacDonald.	City Tech : the first 40 years / by...							

Figure 7
Broome Community College replacement OPAC for Aleph 500.

So I wouldn't try API programming to replace the staff client. But the API worked great for replacing the Aleph Web OPAC.⁵

University of Tennessee

Mike Rogers, a computer systems specialist at the University of Tennessee Libraries, provided a response regarding that institution's experiences with the ALEPH 500 system from Ex Libris:

The University of Tennessee Library uses Aleph 500 as our primary ILS system. Apart from the "out-of-the-box" system (which for us includes the Cataloging, Acquisitions/Serials, Circulation, and Course Reserves modules) we have successfully programmed a number of custom services to meet the needs of students, faculty, and staff.

In general, I have found the Aleph system to be a very open and flexible system. With Aleph, it is possible to embed custom programs within the individual GUI modules and configure them so that staff can run them on an as-needed basis. It is even possible to define permissions so that certain staff can run them. Aleph also has a scheduler which can run the custom programs on a regular basis (hourly, two or more times per day, weekly, monthly, etc.).

An example of Aleph's openness: Aleph utilizes an Oracle database which contains a significant amount of data. In lieu of purchasing third-party reporting software, I have written a number of SQL queries that extract data from the various tables. Depending on the type of report, these custom reports are embedded within the GUI modules and serve as sort of a home-grown reporting center. Another example involves custom Perl and PHP programs. These

too can be embedded within the GUI modules and configured so that only certain staff members have access to run them. In summary, we have a need for our back-end users to run SQL queries, Perl/PHP programs, etc. but we do not want to allow the users to have direct access to the server. However, the Aleph "custom services" option gives us the ability to meet that need.

Another example of custom programming in Aleph involves a set of RSS feeds for new items:

I wrote these with SQL, which writes the output as XML in RSS format.

The XML is converted to HTML using a PHP script. All of this runs automatically once a week on our Aleph server.

Another example of custom programming is a recent service developed for our offsite storage facility. In Aleph, we took advantage of some OPAC functionality that allows patrons to place requests for items. One request that patrons can submit is a "photocopy" of an item. Given Aleph's flexibility, we converted the Photocopy Request functionality into a PDF scan request. This way, patrons can place requests via the Web for PDF scans of items in storage. Some additional programming (shell scripts, SQL, and PHP) automatically sends the patrons a PDF attachment via email once the files are placed on the server.

Additional programming alerts patrons of items not found or items having incomplete/bad citations.

We have also been successful in programming Aleph to extract/manipulate data that can in turn be uploaded to the university's financial system (SAP/IRIS) and the bursar system. Aleph also integrates well with our binding system (ABLE).

I feel in many ways that we have done quite a bit with programming Aleph to meet our needs, but then again feel like we are only scratching the surface. Some things we are considering for the future include sending SMS text message notifications to patrons, utilizing the Aleph X-server for Web services, adding dynamic stacks locator maps for items in the main library, etc.⁶

National Library of Australia

The National Library of Australia, an Ex Libris Voyager site, gave the following responses:⁷

Do you feel like you can pretty much do anything you want with the system, or do you feel constrained?

Very constrained.

Are the APIs offered able to address all the data and functionality within the ILS?

There are no real APIs with current version of the ILS that we are on.

Do you feel like your ILS is too closed?

Yes, definitely too closed.

Do you find the APIs offered by the developer of the ILS to be well documented?

Again, there are no real APIs with current version of the ILS that we are on. With latest versions there are some APIs/Web services, but the documentation is poor.

What programming languages or other tools were you able to use to take advantage of these APIs?

As no real API is available, what work we have done has been connecting straight to the Oracle database using Perl, screen scraping or using its bulk program to wrestle MARC records

What's on your wish list? What kind of APIs would you like to see incorporated into your current or next ILS?

On our wish list would be an API (Web services or other similar technology) that exposes the full functionality of the ILS system. Everything that the OPAC shows you, everything that the call slip client lets you do, everything you can do with the cataloguing client, should be able to be performed using the API as well. The system should expose as its API the exact same interfaces used by its own modules.

Evergreen

Evergreen is an open source ILS developed originally for the PINES consortium of public libraries in Georgia. Following its successful implementation in Georgia, it has attracted the interest of many other public library consortia. The original developers of Evergreen launched a company named Equinox Software in 2007 devoted to its ongoing development, marketing, and support.

Evergreen
www.open-ils.org

While it has also seen a few implementations by single libraries, the majority of interest in Evergreen has been through consortial efforts. In addition to the original PINES consortium, some of the consortia using Evergreen now include the British Columbia SITKA consortium, Evergreen Indiana, SC Lends in South Carolina, and Michigan Library Consortium, with others expected to come online in the near future. The first group of academic

libraries, the Conifer consortium in Ontario, Canada, began production use of Evergreen in August 2009.

Evergreen, given its status as the most recently created library automation system available, is one of the few systems created in the area of service-oriented software. One of the key components of Evergreen is an underlying services layer called OpenSRF, based on Jabber as the messaging protocol. This design makes the software inherently more able to support APIs.

Since Evergreen is an open source ILS, developers have the option of working directly with the source code of the software rather than programming through an abstract API layer. In a service-oriented application like Evergreen, this distinction is more subtle than with proprietary applications, where the only avenue for interacting with the software is through an API. Because the application is built on a services framework, application programming primarily involves coding against the OpenSRF API.

The libraries that have adopted Evergreen so far are not necessarily the types of libraries that have a high demand for APIs. The APIs provide a good foundation for the forward development of the system but tend not to be used by the end-user libraries.

Equinox Software, the primary firm providing support for Evergreen, fielded the survey questions.

Equinox Software
www.esilibrary.com

Survey Responses⁸

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

The basic programming of the Evergreen application takes place through calls made to the underlying OpenSRF API. This API provides access to all of the underlying data structures.

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

All Evergreen APIs are available at no cost out of the box.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

We document the APIs used by Evergreen, all of which are available for use by third-party developers, on the project wiki, in the Evergreen documentation project, and in blog posts. In addition to wiki-based, blog, and traditional documentation of the various APIs, there is a built-in introspection service for the core API.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

There are some portions of the API available through a RESTful interface, and the entire API is available through XML-RPC. There are also Evergreen-specific libraries which implement http and XMPP bindings for the full API.

Describe the technical architecture of the API.

Evergreen is constructed as a set of loosely coupled services which consume standard data object and provide business logic for implementing specific functions which are required for library workflow.

At the heart of Evergreen is the OpenSRF (Open Service Request Framework) network, a programming language agnostic communication backbone that provides an abstraction layer for inter-service service requests. In addition, OpenSRF provides transparent Load Balancing and High-Availability to an application such as Evergreen.

Adding processing capacity to an application built on OpenSRF is as simple as procuring and provisioning an additional low-cost commodity server, and configuring it to connect to an existing OpenSRF network.

By providing a simple internal API for use by application developers, adding and changing functionality of an OpenSRF application is as simple as defining the input, output and logic of the desired method. There need be no consideration made of potential network topologies, client programming languages, session and transaction semantics, or even external integration.

All of these concerns are handled by OpenSRF itself, and the developer of an OpenSRF application can focus strictly on the specific task at hand.

Additionally, by using OpenSRF, Evergreen can be made to scale from the very tiny—say, a rural, single branch library with less than 10,000 items—to the enormous. No code changes are necessary; only a single configuration file specifying which services to run on each node in an OpenSRF network.

Are your APIs able to perform the following types of transactions? (I've provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Modification of bibliographic records is performed atomically, by updating the record as a whole.

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.;

ability for external applications to execute circulation transactions for checkouts, returns, holds, etc.

The API provides methods for both aggregate and individual availability information for items, and can be used to perform all transactions involving items.

Patron records—personal details, demographic data, authentication credentials, materials charged or re-quested, etc.; real-time synchronization with external authoritative repositories of user populations.

Evergreen can perform batch imports and updates of user records from an external, authoritative source, but does not directly integrate with such sources for real-time authentication.

Circulation transactions/history—detailed usage data for collections, user categories, etc.

Evergreen allows retention of full circulation data, as well as supporting the retention of circulation data stripped off of personally identifiable patron information. All retained data can be explored and reported on using standard database techniques including use of the built-in reporting engine.

Acquisitions and financial data—ability to interact with external accounting or ERP systems.

While acquisitions functionality is currently under development, Evergreen does currently support interaction with external systems through EDI (EDIFACT) file interchange.

Can the API create and modify data, or is it read-only?

The OpenSRF provides access to all functionality defined in the application.

Describe any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

Using the Evergreen APIs, production customers built various tools that fall outside the scope of an ILS which leverages their data. Among others, there are contributed, non-core projects for automated outbound telephony, captive portal authentication and a standalone new-books list. All of these use existing Evergreen API calls with no new code required on the ILS side.

Customers have also created entire customized OPACs to replace the stock OPAC that ships with Evergreen.

In what way do you consider your system to embrace the service-oriented architecture?

In order to create a scalable solution, Evergreen was designed from the beginning as a loose federation of cooperating services that depend on one another and create a fabric of methods that are programming language, location, data, and server independent. Each

service provides a specific set of functionality, and each can be tuned, modified or upgraded without affecting the rest of the system.

This is the very heart and spirit of SOA—that a loosely bound group of services depending on one another for specialized tasks is greater than the sum of its parts. This is a technical, founding principal we use in developing Evergreen.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were submitted from libraries that have implemented Evergreen that make use of its API.

The Library Corporation

The Library Corporation (TLC) offers two different ILS products, with a new system in development. Carl.X finds use in large library organizations, especially municipal libraries and consortia; Library.Solution is a popular choice for small to medium-sized public libraries. Both systems have been implemented primarily in public libraries. A specialized version of Library.Solution has been developed for K-12 school districts. Special and academic libraries represent a small portion of TLC's customer base.

The Library Corporation
www.tlcdelivers.com

Carl X has a long legacy in the library automation arena, tracing its roots to at least the early 1980s. The software was designed originally to operate on Tandem hardware, noted for its extremely high reliability. Carl finds use primarily in large-scale implementations, including consortia and municipal systems. The Library Corporation acquired the software in August 2000 and has evolved the system into Carl.X, which relies on a new technology platform based on UNIX and Oracle. As a system implemented by large library organizations, it fits within the realm where APIs and Web services have a high level of interest.

Library.Solution was developed by The Library Corporation and introduced in 1997. This system was designed for small to medium-large public libraries. As a system that appeals to libraries with limited resources, Library.Solution does not fall within the profile of products where APIs would be in great demand.

TLC has begun development of a new ILS platform named LS2. The initial module of the system, the LS2

PAC, can be used as a public interface with either Carl.X or Library.Solution. LS2, consistent with modern programming practices, has been created in a more service-oriented model and will have more capabilities for delivering APIs that can be accessed by customer libraries than was possible with the company's legacy products. The development of the LS2 platform transitions from legacy products to modern service-oriented software; the LS2 PAC, used in conjunction with TLC's older ILS products, makes their data and services more available through Web services.

Survey Responses⁹

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

Both of our integrated library systems, Library.Solution and CARL.X, are exposed primarily through the LS2 platform, which was designed specifically with access to open APIs in mind. Up to that point, we had a variety of examples where we made data available through these ILS systems, either with old-fashioned protocol methods such as NCIP, Z39.50, or SIP2; through on-demand API required by vendors such as AquaBrowser, where we made our bibliographic data, shelf status, and various items in the catalog available; and finally (and most extensively), for our CARL.X system, Chicago Public Library used API for its public catalog and acquisitions/selections interfaces, all via Web services with different front ends designed by third-party companies.

Of course, we would also include our links to multiple other vendors—Student Information Systems (over 30 brands), EnvisionWare, Tech Logic, Unique Management, Talking Tech, NoveList, Cognos, Syndetic Solutions—which allow integration in both directions between our systems and theirs for sharing or exporting/importing data.

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

We have only provided API access extensively for CARL.X. The business model for open APIs with LS2 has not yet been used. However, that doesn't mean we don't have a business model for openly retrieving and using this client data from our traditional systems. Because we offer flexibility in our public catalog interface with open access to the HTML and the services within it, we had libraries linking up to book jackets from Amazon.com in 1999. We've been connecting with about 30 student information systems for the past 10 years through our Library.Solution for Schools product, which updates student information via user-defined fields.

All of our modules include user-defined fields for things such as statistics, book cataloging, serials, acquisitions, and circulation for a greater degree of flexibility in terms

of help and support. Our customers have found that the public catalog can be modified in any way that they choose, all within the limits of the code it's written in. The system has been open, although not necessarily through a large-scale API. In the future, we expect to find libraries that will have a reason to access our API, and they *have* the ability to do that for anything regarding our public catalog and Web circulation in the LS2 API that works with both Library.Solution and CARL.X.

It has been TLC's long-held conviction that our customers' data belongs to them and we'll give them access to it however possible. We look forward to working with libraries to make sure that we balance the need to have an open API with TLC's ability to rapidly innovate the product.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

With CARL.X, we did publish documentation for our API and confined it to licensed customers and third-party developers. We wouldn't give it to an unrelated third party. It's not a shareware product.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

We use SOAP (Simple Object Access Protocol) and the more "modern" JSON (JavaScript Object Notation) in LS2.

Can the API create and modify data, or is it read-only?

The API is CRUD (create read update destroy).

Describe the technical architecture of your API.

We use DWR (Direct Web Remoting) to create JSON. We use Axis 2.0 for the Web services.

Are your APIs able to perform the following types of transactions? (I've provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Available in LS2 via SOAP and JSON.

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.; ability for external applications to execute circulation transactions for checkouts, returns, holds, etc.

Real-time requests for availability are available in LS2 via SOAP and JSON. Ability to execute circulation transactions for checkouts, returns, and holds is offered in LS2 PAC and Circ via SOAP and JSON.

Patron records—personal details, demographic data, authentication credentials, materials charged or requested, etc.; real-time synchronization with external authoritative repositories of user populations.

Personal details are available in LS2 PAC via SOAP and JSON. Authentication credentials materials charged or requested are available in LS2 PAC via SOAP and JSON. Real-time synchronization with external authoritative repositories of user populations in near real time is available with ActiveDirectory.

Circulation transactions/history—detailed usage data for collections, user categories, etc.

No, only via reporting.

Acquisitions and financial data—ability to interact with external accounting or ERP systems.

No.

Vendor records—synchronization with external ERP.

No.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were submitted from libraries that have implemented TLC products that make use of APIs.

Innovative Interfaces

Innovative Interfaces ranks as one of the largest and most long-standing library automation vendors. Its Millennium ILS has been adopted by thousands of libraries throughout the world. Innovative's ILS has been evolving for over twenty years and has made the transition through several generations of technology. The current system, Millennium, was introduced around 1997.

Innovative Interfaces
www.iii.com

Innovative offers a number of APIs to its customers, mostly packaged into discrete product offerings. In general, the company does not offer a one-size-fits-all system, but rather a system where it licenses a customized set of modules and components that meet the needs of each library. With this approach to pricing, each library pays for the functionality it uses, and does not pay for capabilities that might be available that it does not need. Innovative offers a fairly wide selection of APIs, offered as separately licensed components, consistent with its general pricing strategy.

Innovative launched Encore, a new-generation discovery interface in 2006. Encore also provides a new technology platform for Innovative, developed with current-day technologies more amenable to APIs and Web services. Encore also functions as a service layer into Millennium data and functionality.

The customer base of Innovative spans a wide range of types and sizes of libraries. It provides automation for many large and complex libraries. Its products are used in many regions of the globe. Innovative's customer libraries include many large research libraries (thirty-nine ARL members), municipal and large public library systems, and others that fall within the profile of libraries with advanced needs for interoperability and data services.¹⁰ At the same time, Innovative serves a vast number of libraries that expect fully managed turnkey systems and that do not expect to perform local programming. This heterogeneous customer base presents a challenging environment and provides some perspective on the company's business model of offering its APIs as separately licensed options.

Survey Responses¹¹

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

Our practice has been to develop needs-based APIs with direct input/direction from our customers. As a result, we tend not to have theoretical models, but practical implementations that solve specific and real needs. For example, rather than have a "Patron Record" API, we offer a "Patron Update" API, a "Fine Payment" API, and a "My Account" API—we target specific tasks.

Several of our products offering API functionality include Millennium, Encore, Content Pro, and INN-Reach.

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

API sets are optional functions within Millennium. Depending on the initial needs of the particular library, the Millennium system may be initially configured with the appropriate API functions that the library needs. If so, the price of those functions (as with other functionality) is bundled into the initial price. If a library decides to add this functionality to an existing Millennium system, there is an optional fee.

For Encore, the Query API is a module that is priced separately.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

Technical documentation is available to customers and third-party developers. Techdocs, Innovative's Technical Documentation site, provides information to libraries that

have purchased one or more of Innovative’s Web interface products, such as the Patron Update Web Service, Fine Payment Web Service, My Account Web Service, or Millennium Enterprise Backup.

In addition, Innovative has a separate website, Vendordocs, for the purpose of distributing information important to joint development with vendor partners. More importantly, our online documentation provides Perl and/or JavaScript scripts. Our experience has shown that this is the best way for our partners to explore the functionality of the various APIs.

Innovative provides libraries with a downloadable Patron Web Services Development kit.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

All of Innovative’s modern APIs operate through SOAP, and utilize Perl and JavaScript. Legacy APIs use http:// mechanisms.

Can the API create and modify data, or is it read-only?

See specific API descriptions provided below.

Are your APIs able to perform the following types of transactions? (I’ve provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Provided below is a representative set of APIs offered by our products:

Patron API (URL, read-only). Delivers patron information to a requesting service. This URL-based API sends all the patron information to the requester in a single “dump.” Uses might include exporting a URL to a patron photo for printing on a library card, and working with workstation booking and print control software. SSL encryption is an option. The vendor sends an HTML request to a reserved URL on the Innovative system. Inno-

vative returns all the information in the patron record. Examples of use of this API include patron authentication by eBook vendors and the ability to validate a PIN.

My Account Web Service (read-only). The My Account Web Service allows access to patron circulation information about checkouts (including due dates), items available for pickup, status of outstanding holds, booking information, and fine information. This API would allow libraries to integrate information into an enterprise portal, for example.

Patron Update Web Service (read/write). This API allows 3rd parties to create new patron records and update patron information programmatically. Use it to streamline patron registration and upkeep.

For example: patronFields—An array of Patron record fields that can be changed upon update [see figure 8].

Fines Payment Web Service (read/write). This API accesses patron fine information from Millennium. It is primarily used by 3rd party self-check vendors to enable e-commerce functionality.

Innovative’s Fines Payment Web Service provides an API that allows you to access Innovative data and functionality through a website or Web-enabled application. The Fines Payment Web Service follows the standard Web services model, that is, users of the service request data through XML over http or SOAP and the service returns data as an XML-formatted stream of text.

Inventory Express (read-only). Inventory Express communicates with each vendor’s Web service using a SOAP request tailored to that vendor’s specific protocol. The library then receives inventory information and brief records from these vendors in real time. Library staff may interactively choose a vendor with the title in stock during the

Name as Element: patronFields Name as Array: ArrayOfPatronField		
Element	Description	Data Type
fieldTag	Field tag of the identifying Innovative patron record field. The site Millennium System Coordinator can assist in determining the field to use.	String
marcTag	MARC tag of the identifying Innovative patron record field. Applicable only to libraries that use a MARC tag for this purpose.	String
value	The data for patronFields.	String
valueIsBinary	Whether or not the data is in hexadecimal binary format.	Boolean

Figure 8
Details of data elements involved in Innovative Interfaces Patron Update Web Service.

new-order creation process. Once the vendor is chosen, data provided by the vendor is used to create the bibliographic record. Vendors offering this service include Baker & Taylor, Amazon.com, Coutts, Midwest Tape, and BWI.

Item Status API (read-only). The Item Status API permits both Millennium Circulation and the Express Lane product to interface with a third party RFID system. It enables support of the RFID “pad” that allows checkout/checkin of multiple items at once. It also sets the RFID chips appropriately to allow the item to pass through the security gates.

Millennium Enterprise Backup API (read-only). Enterprise Backup API provides “hooks” to prepare the library’s Millennium system data files for backup so that the library can choose a preferred method to backup the files from disk.

Legato NetWorker Client Interface. This product provides support in the Innovative application for the use of third party Legato NetWorker client software. This allows the Innovative server to participate in an enterprise backup scheme using the Legato NetWorker backup software as an alternative to use of the Innovative application’s integrated backup function. It includes the necessary support in the Innovative application and the service of installing the Legato NetWorker client software on the Innovative server.

Veritas NetWorker Client Interface. This provides similar support for Veritas as Legato, which is described above.

The following are interfaces that allow Millennium to interface to external APIs:

LDAP. Innovative’s External Patron Verification product gives Millennium patrons the ability to self-validate on the external LDAP server. This product is designed to permit the system to first verify the patron against a more authoritative database for the institution, and then if the patron is in good standing with the institution, to continue with the transaction within the Millennium system and the patron’s record. Patron data held within Millennium can be accessed by third-party systems using URL-based or Web services-based APIs.

Single Sign-on. Innovative’s Single Sign-on product allows patrons to use authenticated services hosted by various servers on campus while requiring they authenticate only once per session. This eliminates the need to constantly re-key credentials as different participating servers are accessed.

SUSHI. Millennium Electronic Resource Management includes a configurable Web service, which connects to specified vendors on a library-determined schedule to harvest usage statistics provided in the COUNTER format. This harvesting process follows the SUSHI protocol, and both SUSHI 0.1, 1.0 and 1.6 compliant statistics may be harvested.

Encore APIs

Encore Query API. The Encore Query API is a Web service for libraries that want to give their programming staff hands-on control of its own real-time library data. It is a flexible and robust center for creating your own dynamic front-end application interface for data collection and dissemination, or whatever practical applications you envision. Cataloging data (e.g., bibliographic, holding, item, order, checkin) are made accessible through the Encore Query API. This powerful XML-based API can also access Program Registration and Innovative Electronic Resource Management records.

PLANNED—Reporter Statistics. Reporter API: Direct access to the star-schema data mart that powers the Encore Reporter using an industry standard tool such as Mondrian will allow libraries with technical staff to pull data and create locally authored reports. API access may be used instead of, or in addition to, the Encore Reporter application.

PLANNED—OAI-PMH Server. OAI-PMH Server will allow approved third-party client applications, such as Google or OAIster, to crawl the library’s bibliographic database and harvest the metadata as needed on a library specified schedule. This functionality will mirror that currently available with both the Content Pro and Symposia products.

Content Pro API. OAI-PMH Server: Content Pro and Symposia are data providers, and as such, are fully compliant with the current OAI Protocol for Metadata Harvesting. Service providers (e.g. OAIster, Google) can harvest metadata presented in the expected format.

NCIP. The INN-Reach Consortial Borrowing system employs NCIP to transmit circulation transactions to participating non-Millennium ILS systems.

Describe any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

RFID integration with Item Status API. We have several instances of the use of Item Status API.

- Jefferson County Public Library, Lakewood, CO
- Poudre River Public Library District, Fort Collins, CO
- Mountain View Public Library, Mountainview, CA

Item Status.

*Patron Update Web Service.*¹²

Information on Item Status API

<http://brewing.iii.com/2008/09/10/item-status-api-now-working-with-3m-and-envisionware-rfid-systems>

Patron Update Web service article

www.iii.com/news/it/it_2009_03.pdf

In what way do you consider your system to embrace the service-oriented architecture (SOA)?

Innovative leverages SOA for the development of new products and for the enhancement of existing products. Our applications leverage SOA using multiple back-end infrastructure components which provide encapsulation, code reuse, and deployment flexibility. In addition we are able to leverage these services for both our own application presentation layer as well as customer-facing APIs. For example, with Encore, using SOA has resulted in an implementation that is distributed, easier to manage, and more extensible and reusable for future development.

Are there any other issues regarding APIs that you would like me to be aware of as I develop this report? Are there other approaches that your company supports instead or in addition to APIs for providing end-user access to system data and functionality?

As mentioned earlier, Innovative practices a “needs based” approach to this type of functionality. As libraries identify areas where there are information needs that can be supported by the ILS, we will develop interfaces that support those needs. This also extends to technologies. We implement technologies based on those needs. Whether it results in the implementation of an API or other communication technique, we implement in the manner that best suits the needs of the problem to be solved.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were submitted from libraries that have implemented Innovative Interfaces products that make use of APIs.

Koha

The open source Koha ILS has attracted a number of libraries of many different types in many regions of the world. In the United States, Koha has been adopted by a mix of public and academic libraries, primarily in the small to mid-sized range. The system was originally developed in New Zealand in 1999 and has seen continuous development and enhancement in the past decade. In the United States, most libraries implementing Koha have done so through paid service contracts with commercial firms, including LibLime, PTFS, and ByWater Solutions. LibLime ranks as the largest of these firms, has been providing these services the longest, and has the most client libraries. Given its vintage of development, Koha predates the period when most applications were built according to a service-oriented architecture. The software has evolved significantly since its original version. The ILS has become more sophisticated in functionality and has been extended to support a wider range of standards; a limited number of APIs have been added.

Koha finds use primarily in smaller libraries that do not necessarily expect to work with their ILS through an API. A large portion of the libraries using Koha rely on a support company for all the technical work involved in implementation, and many take advantage hosting services.

As the dominant company involved with providing support services, LibLime provided responses to the survey questions regarding Koha’s support for APIs.

Survey Responses¹³

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

LibLime created and maintains a simple set of RESTful APIs that enable our customers to write applications to interact with their Koha databases. The same set of APIs are implemented in our ‡biblios.net product and are documented at the ‡biblios.net Web services website.

‡biblios.net Web services website

<https://bws.biblios.net/doku.php>

Koha Web Services Overview

http://wiki.koha.org/doku.php?id=en:development:web_services

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

Access is included with the base product.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

The documentation is available to anyone.

In a follow-up question LibLime was asked to indicate what documentation of the APIs has been developed.

There are some portions of the Community documentation that are sparse, that's for sure. Since Koha is open source, it's assumed that any programmers that need to use the API will just delve into the programming itself.

Note that all of the APIs supported in \ddot{b} iblios.net are also supported natively in Koha, and they share the same codebase.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

It operates through REST Web services.

Can the API create and modify data, or is it read-only?

Data can be created and modified using the API.

Describe the technical architecture of the API.

LibLime's Web services architecture goes beyond the nebulous so-called "Services Oriented Architecture" and focuses instead on resources using the principles embedded in the Resources Oriented Architecture and in Roy Fielding's doctoral dissertation in Representational State Transfer.

Roy Fielding, doctoral dissertation, chapter 5, "Representational State Transfer (REST)"

www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Are your APIs able to perform the following types of transactions? (I've provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Yes.

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.; ability for external applications to execute circulation transactions for check-outs, returns, holds, etc.

Yes.

Patron records—personal details, demographic data, authentication credentials, materials charged or requested, etc.; real-time synchronization with external authoritative repositories of user populations.

Yes.

Circulation transactions/history—detailed usage data for collections, user categories, etc.

Yes.

Acquisitions and financial data—ability to interact with external accounting or ERP systems.

Yes, in theory, though there are no live examples.

Vendor records—synchronization with external ERP

Yes.

Please feel free to mention any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

We're unaware of any customers that even know what an API is. Most of the time we simply provide tools and widgets that access the various APIs embedded in our products and our customers use those tools/widgets. APIs for LibLime are not an add-on afterthought; they are how the product is actually designed.

The most interesting application of our APIs exists on the \ddot{b} iblios.net Web services platform. Quite a few libraries have used Koha's built-in cataloging editor to contribute to the \ddot{b} iblios.net platform, which uses the API even if they don't know it does.

The cataloging editor in \ddot{b} iblios uses the authority Web service to provide an auto-complete dropdown list of authority records in authority controlled fields.

We have a few Koha customers that synchronize patron data with LDAP.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were received from libraries using Koha reflecting their use of APIs or Web services.

Polaris

Polaris, based in Syracuse, New York, focuses on the public library arena. The company offers the Polaris Integrated Library System. The company was originally created as a division of Gaylord, but became an independent company in 2003 when Gaylord was acquired by Demco. Polaris was introduced around 1997 and has steadily increased its customer base ever since. Polaris finds use in public libraries of all sizes; in recent years, it has found inroads into the municipal library arena with sales to Phoenix Public Library and Miami Public Library. From its beginning, Polaris has been designed to accommodate large

Polaris
www.polarislibrary.com

libraries. With this emphasis on larger libraries comes an expectation for the extensibility offered through APIs.

Survey Responses¹⁴

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

The Polaris API needs to be viewed in context with the underlying architecture of the Polaris ILS. The Polaris ILS is an open database. Polaris customers have a fully documented copy of the database schema and, with knowledge of SQL, can freely access and leverage the database without permission or assistance from Polaris.

The Polaris API simply enables customers to take this level of access and control a step further by ensuring that any enhancements or procedures written against the Polaris ILS are automatically updated against any changes to Polaris's stored procedures. Essentially, as the Polaris ILS evolves with every release, so do the enhancements created by the customer.

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

The API is an extra-cost option due to the level of ongoing support needed to maintain the product.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

Yes, Polaris does publish documentation regarding the API for our customers. Customers are free to share that documentation with third-party developers at their discretion.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

The API uses standard database communication mechanisms, such as JDBC, ODBC, OLEDB, ADO, etc. Polaris chose this approach because they are existing standards. In addition, this approach is more secure and better protects the integrity of the customer's data, as you need to be inside the customer's network to access the database.

However, Polaris does offer a SOAP-based holdings Web service that provides item level call number, location, availability and real-time status information. The Web service is provided at no cost as part of the PAC Web application and includes documentation on its use and example service calls.

Can the API create and modify data, or is it read-only?

Yes, the Polaris API enables the customer to create and modify data.

Please describe the technical architecture of your APIs.

The Polaris API is a database of stored procedures exposed through standard database communication mechanisms including JDBC, ODBC, OLEDB, ADO, etc.

Are your APIs able to perform the following types of transactions? (I've provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Polaris provides the capability for complete and flexible extraction of bibliographic content through the Simply Reports Export Express product, not through the API.

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.

Yes, the API is able to perform this type of transaction.

Ability for external applications to execute circulation transactions for checkouts, returns, holds, etc.

The API does address renewals and holds, but does not address returns and checkouts. However, customers can use SIP to accomplish these types of transactions.

Patron records—personal details, demographic data, authentication credentials, materials charged or requested, etc.

Yes. The API does perform these types of transactions.

Real-time synchronization with external authoritative repositories of user populations.

This can be accomplished without the API. Polaris has a utility that interfaces with Civic Technologies .

Circulation transactions/history—detailed usage data for collections, user categories, etc.

This can be accomplished without the API by querying the database through the use of Simply Reports or SQL Reporting Services.

Acquisitions and financial data—ability to interact with external accounting or ERP systems.

Polaris has developed a utility for interaction with external financial systems through a generic XML-based data interchange. The adaptability of this model is dependent on the capabilities of the financial system with regards to data exchange with external systems. This interface has been integrated at Dallas Public Library with the Advanantage 3 Financial System.

Vendor records—synchronization with external ERP.

No, the API is not able to perform this type of transaction.

Please feel free to mention any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

Phoenix Public Library has been able to interface with Endeca for the PAC.

Also, Henderson District Public Libraries in Nevada has leveraged our open database, without the need for an API, for quite some time (they also have our API). As one example, Henderson wrote their own .NET application that uses the Polaris database and the Stamps.com API to generate overdue and hold notices in postcard format. The notices, including postage, are printed in-house by the circulation department and mailed. Patron accounts are updated automatically. In addition to reducing staff time, notices are turned around faster, and the library saved significantly on printing and postage.

In what way do you consider your system to embrace the service-oriented architecture?

Polaris offers a SOAP-based holdings Web service that provides item level call number, location, availability and real-time status information. The Web service is provided at no cost as part of the PAC Web application and includes documentation on its use and example service calls.

Whenever possible, any integration with third party content providers is done through Web services published by the vendors. Polaris currently consumes Web services from NoveList Select, Baker & Taylor's Content Café, and Syndetics, as well as Baker & Taylor, Ingram, BWI & United Library Services for Titles to Go.

Are there any other issues regarding APIs that you would like me to be aware of as I develop this report? Are there other approaches that your company supports instead or in addition to APIs for providing end-user access to system data and functionality?

Polaris offers tools that enable customers of all skill levels to access their data. For library staff with technical skills, the Polaris API is easy to use, intuitive and supplements the "openness" that is built into the fabric of the system.

For customers who do not have this level of technical skill, or who are not interested in devoting staff resources to these types of projects, Polaris offers tools to make it easy to access the data and interface with other vendors, without the customer needing a programming background. As noted above, much of the functionality you inquired about can be accomplished without the need for an API or advanced programming skills on the part of our customers.

As an additional example, SimplyReports, Polaris's Web-based reporting tool, enables customers to build hundreds of thousands of custom reports using a GUI interface. With SimplyReports, customers point and click to the data elements they are interested in—requiring no programming skills or knowledge of SQL. SimplyReports

can be supplemented with Export Express, Polaris's export utility, to enable users to easily extract records from the database and save them in exportable format.

Essentially, Polaris strives to provide a balance for our user community—an open system, but one that is fully functional and enables our customers to focusing on meeting the needs of their communities.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were received from libraries using Polaris reflecting their use of APIs or Web services.

SirsiDynix

SirsiDynix, a consolidated company formed through a series of business acquisitions, offers Symphony as its primary ILS. The company also continues to support Horizon and Dynix Classic. SirsiDynix has library customers in many regions of the globe spanning many different library types. Its customers run the gamut of large research libraries to small public libraries. The company faces the need to provide powerful flexible tools to its larger customers that have more complex automation needs while delivering fully managed self-contained systems for others. One of the key business strategies involves an emphasis on software-as-a-service (SaaS). SaaS is quite consistent with the use of APIs, but tends to be implemented by libraries of smaller size and complexity, which are less likely to take advantage of APIs.

SirsiDynix
www.sirsidynix.com

Symphony, formerly known as Unicorn, has been steadily evolving since it was introduced in the early 1980s. SirsiDynix (then Sirsi Corporation) became the first ILS vendor to enable its customer libraries access to its internal API in late 1995. This API offered comprehensive access to all the data and functions of the underlying Unicorn system. It operated at the UNIX command line and generally returned results as pipe-delimited text.

SirsiDynix offers training for its customers that want to use the API. While the company charges for this training and generally prefers that libraries not use the API unless they have taken this training, there is no additional license fee involved with the API itself.

Since the Unicorn API allows libraries to create and modify data records, those making use of it must work with great skill and care to ensure that no databases are

damaged or corrupted. Use of the API can result in support questions of great complexity. The API gives the library programmer great power to work with the system; with that power comes the responsibility not to allow an error in a custom script to wreak havoc on the library's data.

Survey Responses¹⁵

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

SirsiDynix provides multiple APIs with our various products. We provide the following:

Symphony APIs. Approximately 415 API commands into the Symphony Application server. All of the SirsiDynix clients (Java Workflows, e-Library, StaffWeb) use these same APIs that are available to the customer.

SIP APIs. We fully support SIP and provide entry points into our SIP Server

NCIP APIs. We fully support NCIP and provide entry points into our NCIP Server

Symphony Web Services. We provide a set of Web service (SOAP and RESTful) methods to the Symphony Server

Discovery Platform Web Services. We provide a set of Web services (SOAP today and RESTful shortly) into the Enterprise Platform.

The Horizon and Dynix products provide full and direct access to the database, so many customers use standard SQL tools to interact directly with the database.

The Symphony product provides either an embedded database or a full use database, depending on customer preferences and pricing. With a full use database, the customer also has direct access to the data in the database and can use standard SQL tools to interact with the database.

Summary of APIs and Web services:

SIP API—Industry standard for self-service circulation used by Horizon and Symphony

- Supports 15 standard SIP messages
- Supports 5 proprietary SIP messages

NCIP API—Next generation of SIP

- Supports 15 standard messages

Symphony API (HAT Protocol)—Full access to Symphony

- 415 supported HAT commands

Symphony Web Services—Simplification of HAT API

- Version 1: Supports 30–40 Web service methods

- Version 2: Roadmap—target mobile, e-Library and JWF APIs

Discovery Platform Web Services—Used by Enterprise, SchoolRooms and Hyperion applications

- 10 Web services (WSDLs): 146 methods

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

All APIs and Web services are automatically installed and available to our customers. Customers are not required to purchase our training subscription but are strongly encouraged to do so. It is an extra optional cost that provides technical documentation, hands-on lab experience, source samples, etc.

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

Yes, we publish hundreds and hundreds of pages of technical documentation and descriptions for our APIs and Web services.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

Both. Our Symphony API is a proprietary protocol utilizing TCP/IP sockets and formatted strings for all of the commands, parameters, and return values. Our Web services are implemented with both SOAP and REST.

Can the API create and modify data, or is it read-only?

All APIs and Web services can modify data and also retrieve read-only data.

Describe the technical architecture of your APIs.

The Symphony APIs (proprietary protocol) provide access to 100% of the back end Symphony functionality. Our own client applications utilize this protocol and these APIs. The Symphony Web Service is architected to be more modular. We are focusing extensive research and design effort into grouping together related Web service calls into a Web service and then providing multiple Web services (WSDLs) that provide more meaningful functional separation. These individual Web services provide a true SOA. The Enterprise Discovery Platform Web services consist of 10 distinct services with our release of Enterprise V3. The Enterprise patron user interface and the Enterprise administrative user interface utilize these 10 Web services exclusively. These are the same Web services we'll release to our customers.

The following picture [see figure 9] shows this architecture.

Are your APIs able to perform the following types of transactions? (I've provided some examples of some

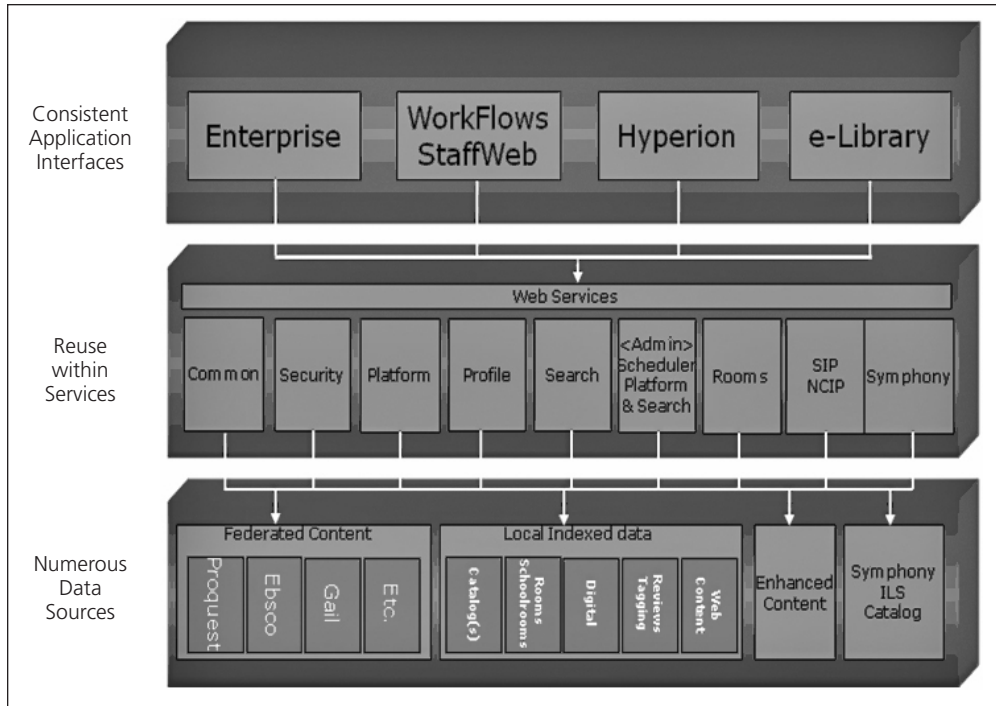


Figure 9
SirsiDynix technical architecture.

specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Yes.

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.; ability for external applications to execute circulation transactions for check-outs, returns, holds, etc.

Yes.

Patron records—personal details, demographic data, authentication credentials, materials charged or re-quested, etc.; real-time synchronization with external authoritative repositories of user populations.

Yes.

Circulation transactions/history—detailed usage data for collections, user categories, etc

Yes.

Acquisitions and financial data—ability to interact with external accounting or ERP systems

Yes, we can retrieve vendor records and also have customers who have used our API to synchronize and external ERP with the records in the Symphony database.

Vendor records—synchronization with external ERP.

Yes, we can retrieve vendor records and also have customers who have used our API to synchronize and external ERP with the records in the Symphony database.

Describe any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

We've had numerous customers utilize our APIs to write client applications and also use scripting tools to automate their various unique business practices.

In what way do you consider your system to embrace the service-oriented architecture?

A true SOA provides distinct services that are well defined, documented, and easy to use. With our new Web services and architecture, development tools can consume our WSDLs and create stub implementations to our services within minutes. We are now soliciting the help of our customer base, vendors, and others thru our Strategic Partner Program. The partners will be helping us define additional services and also to enhance our existing services. Our Enterprise V3 product was 100% written based on the Discovery Web Services. We spent a tremendous amount of time designing these services with SOA in mind.

Are there any other issues regarding APIs that you would like me to be aware of as I develop this report? Are there other approaches that your company supports instead

or in addition to APIs for providing end-user access to system data and functionality?

As mentioned above, direct database access is available to our Horizon/Dynix customers. Direct database access is also a possibility with our Symphony product depending on the licensing arrangement with the customer.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were received from libraries using Sirsi-Dynix reflecting their use of APIs or Web services.

Talis

Talis, a library automation company based in Birmingham, United Kingdom, offers the Talis Library Management Suite (LMS), used exclusively in the United Kingdom. While the company focuses its business activities primarily in the United Kingdom, it has been a proponent of the service-oriented architecture and semantic Web technologies to a much wider audience. Representatives from Talis have been actively involved in conferences and at other venues outside the United Kingdom, especially in the United States, promoting Web 2.0 technologies, SOA, and the semantic Web. The company has been involved in providing open source tools while it continues to license proprietary software.

Talis
www.talis.com

Alto, the company's primary integrated library system (or library management system in UK parlance) has been adopted primarily by public and academic libraries. The Alto automation system has evolved steadily over a long history of development. While Alto itself was created prior to the age of APIs, Web services, and SOA, the company has developed a strategy of creating integration products that layer APIs in the form of Web services onto existing traditional products.

In recent years, Talis has increasingly defined its technology strategy around the service-oriented architecture. The company continues to develop and support its traditional products and services, but has produced a new set of products that rely on SOA to connect library products with other institutional technology infrastructure components.

The LMS marketplace in the United Kingdom, where Talis does business, presents very limited opportunities for major new sales. This finite body of libraries offers only a handful of LMS procurements each year. Given

these limitations, Talis supplements its efforts on traditional library software development with the creation of an SOA framework that can be used in the creation of integration products. Within the library market, these tools might provide interoperability between Alto and the business systems of the local authorities or a university. Ambitiously, Talis also sees its integration products finding use with its competitors' LMS products. Taken more broadly, Talis sees its SOA framework as having the capability to address other business scenarios where interoperability can be accomplished through Web services.

As noted below, Talis offers a set of licensed, commercial quality products as modules of a broader platform called Keystone. Talis is also involved in a project called Jangle, a specification of a middleware layer that can be applied to any ILS to expose a standard set of services through the Atom Publishing Protocol. Some open source implementations have been built around the Jangle specification. The basic idea involves building a core middleware layer that can be associated with any ILS in order to provide a standard set of services as Atom feeds. The key to implementing Jangle involves creating a connector between the proprietary or idiosyncratic APIs of any given ILS to the Jangle core. Talis staff have been involved in the creation of the Jangle specification and in open source projects to implement it. Yet the company does not assert ownership of the project, but rather promotes involvement of a broader community of open source developers. Jangle, for example, has been positioned as one of the methods for providing connections between the open source VuFind discovery interface and an ILS.

Jangle
<http://code.google.com/p/jangle>

Survey Responses

Author's Note: Talis did not provide direct answers to the questions, but submitted a draft of a white paper that addresses the topic. Excerpts of this paper have been selected in response to the survey questions.

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

"Talis SOA is a continuously evolving architecture which is the technical engine behind three commercial areas.

Talis Keystone is the product which enables institutional integration for both academic institutions and local authorities across multiple Library Management Systems in the areas of portal integration, online payments of library fines and charges, integration with corporate finance systems, inte-

gration with identity systems - such as student registries and IDentity Management (IDM) services and Customer Relationship Management (CRM) systems. Talis Keystone is a fully managed product, backed by the Talis Consulting department who are experienced in providing integration project services to customers enabling them to tailor their integrated solution to meet their exact commercial requirements.

Talis Connect Services is the licence issued to Talis customers to enable them to provide interoperability with third party suppliers who are members of the Talis Additions Partnership Programme. Working closely with our partners to implement an integrated solution on top of the web service stack in the Talis SOA architecture, we at Talis are able to give our customers the assurances that their products fully interoperate with their library management solution.

Talis Local Data Services is the component inside of the Talis Library Management System (Alto) which provides internal interoperability for features inside of the Talis product portfolio. Talis products are using the SOA architecture as the interface to the business logic and services of the core library system; over time more products and services will evolve over to this architecture.”¹⁶

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

“Talis Keystone is a licensed product with a central middleware and a series of bolt on modules that can be activated as required, depending on the business requirements of the library service. This gives customers choice and flexibility to create the integration solutions that they require, rather than imposing a suite of solutions which may not be entirely relevant. As well as the middleware license, each module also has a license fee. Talis Connect Services has a similar business model to Talis Keystone; however, Talis Local Data Services is part of the Talis Alto (Library Management System) subscription.”¹⁷

Do you publish documentation for your APIs? Is it available only to licensed customers, or to third-party developers as well?

Author’s note: Information both from Talis and customer sites indicates that the company provides extensive documentation for its APIs. Customer comments indicate that only licensed sites receive access to that documentation.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

“The Talis SOA architecture is a standalone appliance which is designed with the principles of componentisation, reusability, interoperability, orchestration and loose coupling at its core. Over time, the use of SOAP based web services has been fully replaced with a growing portfolio of approximately seventy RESTful web services in various functional areas. Based upon the Java platform and on open standards such as XML and http the web services have enabled developers in libraries, in other institutional departments and in third party suppliers to easily integrate their solutions with their library management system.”¹⁸

Can the API create and modify data, or is it read-only?

Author’s note: The APIs described in the white paper include ones that involve transactions that modify library data as well as those that read and report data.

Describe the technical architecture of the API.

“The Talis SOA architecture is a standalone appliance which is designed with the principles of componentisation, reusability, interoperability, orchestration and loose coupling at its core. Over time, the use of SOAP based web services has been fully replaced with a growing portfolio of approximately seventy RESTful web services in various functional areas. Based upon the Java platform and on open standards such as XML and http the web services have enabled developers in libraries, in other institutional departments and in third party suppliers to easily integrate their solutions with their library management system.”¹⁹

Are your APIs able to perform the following types of transactions? (I’ve provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Author’s note: Appendix B of the white paper provides an inventory of the Web services offered by Talis. Business areas addressed include MyAccount, Finance, e-pay, borrower services, circulation, loan services, charge services, message services, item services, and system services.”²⁰

Please feel free to mention any case studies where one of your customers has been able to perform interesting work with your system through the use of APIs.

Author’s note: Talis describes several organizations that have purchased its integration components:

Queens University Belfast licensed Talis Keystone to support e-payments. The project involved using Keystone to connect the Talis LMS with World-Pay, an internet payment solution through the campus portal.²¹

A similar project took place at the University of Wolverhampton which “chose to link directly

from the university web page to the e-Payment provider. Students and staff can discover the amount that they owe to the library and can conveniently pay their fines and charges online. This seamless transaction from the university web page to the e-payment provider is enabled using Talis Keystone to integrate the two systems and provide an enhanced service for students.²²

The University of Manchester, licensed Talis Keystone to integrate their Ex Libris Voyager LMS with their campus portal based on Microsoft SharePoint allowing students to view library information including number of items charged, items overdue, and fines owed.²³

Customer Perspectives and Comments

A systems manager in a library that uses Talis Alto and Keystone offers some perspective of the Talis approach to APIs:

Talis uses a framework called Keystone for providing APIs to the system. You can use either SOAP or REST with Keystone.

Keystone has a “core” module to provide the framework and then you purchase “modules” that provide specific functionality.

We currently have the “View My Account” module installed and we use this to display borrowers’ account details in the campus portal. We are about to install the “Borrower ePayments” module which allows fines and charges to be paid online. One module we would like to install in future is the acquisitions/finance system link so we can send processed invoices to our finance system automatically for payment.

Keystone at the moment certainly does not offer access to all the features of the Talis system. However, this is because Talis have not yet written modules. I suspect this will be all driven by demand—if you are willing to pay for a module, Talis will, I’m sure, be willing to write it. Therefore, I don’t think Talis are trying to keep their system closed.

The APIs are well documented, though we did have an issue of not being able to get the ePayment API doc before we purchased, so it was very difficult to work out exactly what it did and what we were actually buying.

I think you need to have a reasonable level of programming skill to use these APIs, even though the REST interface is pretty straightforward.

Systems librarians without formal programming training and who are perhaps used to making small changes to letter scripts, would probably struggle. I’ve got more of a programming background and I don’t really have the time to spend on it. The “View My

Account” work was done by our portal developers who are web developers. For the “ePayments,” Talis actually wrote a front-end application for another customer who didn’t have a portal and we’re using that.

I’m not sure if this really is SOA—I’ve never been comfortable that I’ve really understood the term. It always seemed a bit of a buzzword. I’m sure Talis would say that this is SOA!

My main concern with Keystone is not anything technical. Because you have to pay for the core and each module plus a heavy annual support fee, it means the cost of these things soon mount and add significantly to the overall cost of the system.

Talis are also developing what they call a light-weight integration protocol called Jangle.

This seems quite broad in scope, both in terms of what it will allow access to in the ILS and in the variety of systems it can support (it appears that they have a Jangle connector for a non-Talis system before they have produced one for Talis!) It will be interesting to see what becomes of this.²⁴

VTLS

VTLS, based in Blacksburg, Virginia, offers the Virtua ILS, which was initially introduced around 1998. One of the longest standing companies in the industry, VTLS has been in continuous operation since 1974.

VTLS

www.vtls.com

A diverse group of libraries have implemented the Virtua ILS, including academic, public, national, and many regional consortia. The customer base of libraries using Virtua is internationally diverse, including the United States, Europe, Asia, Australia, the Middle East, South Asia, and many others. Especially noteworthy among its clientele is the Queens Borough Public Library, the busiest and one of the largest public libraries in the United States.

The company has recently announced Chamo, a new public interface for Virtua based on the open source Drupal extensible content management system. One of the characteristics of Chamo that VTLS emphasizes is an architecture that directly accesses the Oracle database API rather than functioning through the Virtua server application.

VTLS serves many libraries with very complex automation needs and performs a great deal of custom development of its systems. This niche of the library automation arena expects more than off-the-shelf functionality, making support for APIs especially vital.

Survey Responses²⁵

Describe, in general terms, the APIs that your company offers for your primary ILS and other library automation products.

Both APIs and database views to support special needs are available to customers. SOAP is used in VITAL; Chamo's API is a simple XML over http service. It will support SOAP in the next release.

What is the business model for your APIs? Is access to the APIs included with the base product, or is it an extra-cost option?

Database views are free. APIs are licensed to customers that license the primary product (Virtua or VITAL).

Do you publish documentation for your APIs? Is it available only to licensed customers or to third-party developers as well?

Yes, APIs are documented, but available only to customers with an active license.

Does the API operate through REST or SOAP Web services, or does it use proprietary mechanisms for transmitting data and requests?

SOAP is supported in VITAL only. (Note from VTLS: VITAL is VTLS's institutional repository product based on Fedora.)

Can the API create and modify data, or is it read-only?

The APIs allow users to place and modify patron requests and perform renewals. The rest is read-only: search, availability information, and patron information including fines, checkouts, and requests. If users choose to work with Z39.50 and its extensions, then they can access or modify any and all data within the database.

Describe the technical architecture of your APIs

The API is part of the main Chamo application, a Java EE servlet. Since the API is simple XML over http, it uses the same servlet technology as the standard Web interface.

Are your APIs able to perform the following types of transactions? (I've provided some examples of some specific activities that libraries may want to accomplish, but there may be others that you support that you may want to mention.)

Bibliographic records—ability to extract or update metadata on a field-by-field basis, beyond the capabilities of standard protocols such as Z39.50.

Single API call to retrieve all bibliographic data including the full record in MARCXML format, and all item information including availability, and serials issues. There are no smaller or more granular operations. (Our API design is focused on simplicity.)

API calls that update item information:

- Renew a checked out item for a patron.
- Request an item for a patron.
- Modify an existing request
- Cancel a request

Item/holdings records—ability to respond to real-time requests for availability, circulation status, etc.; ability for external applications to execute circulation transactions for check-outs, returns, holds, etc.

Yes—see above.

Patron records—personal details, demographic data, authentication credentials, materials charged or requested, etc.; real-time synchronization with external authoritative repositories of user populations.

There are two API calls here:

- Authenticate a patron via ID and password
- Retrieve all patron details, including contact information, checkouts, requests, and fines.

Circulation transactions/history—detailed usage data for collections, user categories, etc.

This is part of the previous API. There are no APIs for this data except the patron-related ones.

Acquisitions and financial data—ability to interact with external accounting or ERP systems.

Yes, but only available through Edifact.

Vendor records—synchronization with external ERP.

No, not supported.

In what way do you consider your system to embrace the service-oriented architecture?

SOA is implemented in full scale add-on products like FRBR SaaS.

CASE STUDIES AND CUSTOMER RESPONSES

No responses were received from libraries using VTLS reflecting their use of APIs or Web services.

Notes

1. Marshall Breeding, "Ex Libris Sets Strategic Course on Open Systems," *Smart Libraries Newsletter*, Aug. 2008; "Ex Libris Launches Open-Platform Program, Reaffirming Its Commitment to Openness as a Core Company Value," Ex Libris press release, June 29, 2008, www.librarytechnology.org/ltg-displaytext.pl?RC=13372 (accessed Oct. 14, 2009).

2. Tamar Sedeh, telephone interview by the author, Aug. 13, 2009.
3. Tamar Sadeh, e-mail and telephone interview by the author, including a Web demo of the EL Commons, Aug. 13, 2009.
4. Ken Mitchell, e-mail interview by the author, Aug. 31, 2009.
5. Mike Curtis, e-mail interview by the author, Aug. 24, 2009.
6. Mike Rogers, e-mail interview by the author, Aug. 20, 2009.
7. ILS manager at National Library of Australia, e-mail interview by the author, July 27, 2009.
8. Brad LaJeunesse, president, Equinox Software, e-mail interview by the author, July 26, 2009.
9. Gar Sydnor, senior vice president, The Library Corporation, e-mail interview by the author, Aug. 6, 2009.
10. Gene Shimshock, vice president of marketing, Innovative Interfaces, e-mail interview by the author, Aug. 13, 2009.
11. Ibid.
12. Innovative Interfaces, "At University of Lethbridge (Canada), Machines Talk When People Talk: Patron Web Services Keeps Library in Synch with Campus," *INN>Touch* 22, no. 4 (March 2009): 4, www.iii.com/news/it/it_2009_03.pdf.
13. Joshua Ferraro, CEO, LibLime, e-mail interview by the author, Aug. 18, 2009.
14. Kathryn Miller, director of marketing, Polaris, e-mail interview by the author, Aug. 14, 2009.
15. Talin Bingham, chief technical officer, SirsiDynix, e-mail interview by the author, Aug. 28, 2009.
16. Andy Latham, "Opening the Wall of the Library: SOA and Web Services at Talis" (Aug. 2009): 3, www.talis.com/integration/documents/talis_SOA_white_paper_august_2009.pdf (accessed Aug. 30, 2009).
17. Ibid., 4.
18. Ibid.
19. Ibid.
20. Ibid, 21-23.
21. Ibid, 20; see also "Students at Queen's University Belfast Can Now Pay Their Library Fine Online with Debit and Credit Cards," Talis press release, Jan. 10, 2008, www.librarytechnology.org/ltg-displaytext.pl?RC=12971 (accessed Oct. 1, 2009).
22. "Online Payments for Students at the University of Wolverhampton," Talis press release, Dec. 4, 2008, www.librarytechnology.org/ltg-displaytext.pl?RC=13696 (accessed Oct. 1, 2009).
23. Latham, "Opening the Wall," 5.
24. Anonymous [a systems manager in a library that uses Talis Alto and Keystone], e-mail interview by the author, Sept. 1, 2009.
25. Vinod Chachra, president and CEO, VTLS, e-mail interview by the author, Sept. 1, 2009.

STATEMENT OF OWNERSHIP AND MANAGEMENT

Library Technology Reports, Publication No. 024-897, is published 8 times per year by the American Library Association, 50 E. Huron Street, Chicago, IL 60611-2795. It is the official publication of ALA Techsource, a unit of the publishing department of the ALA. Annual subscription price, \$325. American Library Association, 50 E. Huron Street, Chicago, IL 60611-2795, owner and publisher; Dan Freeman, 50 East Huron Street, Chicago, IL 60611-2795, editor. Periodicals postage paid at Chicago, Illinois, and additional mailing offices. Printed in U.S.A. As a nonprofit organization authorized to mail at special rates (Section 423-12, Domestic Mail Manual), the purpose, function, and nonprofit status of this organization and the exempt status for federal income tax purposes has not changed during the preceding twelve months.

EXTENT AND NATURE OF CIRCULATION

("Average" figures denote the average number of copies printed each issue during the previous twelve months. "Actual" figures denote actual numbers of copies of single issue published nearest to the filing date—August/September 2009 issue.) Total number of copies printed: Average, 1,391; Actual, 1,339. Sales through dealers and carriers, street vendors, counter sales, and other paid distribution outside USPS: Average: 110; Actual: 105. Total paid and/or requested circulation: Average, 796; Actual, 759. Free distribution by mail, carrier, or other means, samples, complimentary and other free copies: Average, 0; Actual, 0. Total distribution: Average, 1,006; Actual, 961. Copies not distributed: office use, leftover, unaccounted, spoiled after printing: Average, 385; Actual, 378. Total (previous two entries): Average, 1,391; Actual, 1,339.