Chapter 3

Observations and Conclusions

API Hype and Reality

Abstract

This chapter of "Opening up Library Systems through Web Services and SOA: Hype or Reality" examines the general conclusions that can be drawn from the data developed in this report. The evidence gathered in this report reflects ongoing progress toward more openness in library automation systems, but also that much work remains. We see a variety of options and opportunities. Libraries that expect to work with their automation system as delivered and not become involved in local extensions or programming will find that the majority of systems were built for that kind of use. For libraries that want to do more with their automation systems, however, we see a great deal of functionality possible today through open interfaces, with momentum toward creating much more.

In this report, we have seen a wide variety of approaches to the way that APIs can be incorporated into integrated library systems. Many examples demonstrate the benefits that libraries gain through access to a well-developed API for their ILS. We have also noted that this capability appeals to only a relatively narrow niche of libraries. The majority of libraries expect the ILS, an entity in which they have invested significant resources, to deliver the functionality they require as-is without the need for them to perform local programming. APIs find their most enthusiastic adherents in larger libraries with complex automation requirements. As the general software arena gravitates more toward the service-oriented architecture, it will become increasingly important for all library automation products to evolve accordingly.

In broad terms, we found no glaring inconsistencies between the claims made by vendors for opening up their systems through APIs and the capabilities actually delivered. APIs that function as important tools that find use in strategic library projects have been created and documented, particularly in the ILS products used by large academic and municipal libraries. Yet even with those that place the highest emphasis on exposing their systems through Web services and other APIs, many gaps remain in areas not yet addressed.

We also note that the two open source systems lag behind proprietary systems in terms of customer-facing APIs that result in tangible activities which extend functionality or enable interoperability. While the open source model may offer many other advantages, we see fewer APIs designed for library customer use and a much lower level of activity among libraries executing projects that make use of this approach. This trend has much to do with the demographics of the libraries using the software.

Developers of open source ILS products find these programs in an early level of maturity and continue to focus on expanding the core functionality to meet basic expectations. By and large, the kinds of libraries that would make use of an open API have not adopted open source ILS products.

Although we see many ILS products that offer extensive APIs, we found no products that meet the ideal of comprehensive access to data and functionality through an open API. Even those with the most advanced APIs have work left to do in the quest toward fully open systems.

Having collected information from and about each of this slate of companies and products, we can offer some general observations.

Ex Libris makes the strongest claims toward open systems. It promotes its Open Platform Program as one of

its key business and technology strategies. To a very large extent, we see substance behind the program and that Ex Libris has put more tools in the hands of its customers than have its competitors. Yet we received comments from its customers indicating that gaps remain. Some aspects of functionality have not yet been addressed by the current set of APIs. The Open Platform Program was launched recently; the success of the program will be marked by the delivery of more APIs, more documentation, more consistency among the APIs, especially in libraries becoming more engaged in their use. While each of Ex Libris' products offers APIs in some way and in varying levels of scope, the company's concerted effort to rework the APIs of each of its separate products into a more unified and comprehensive body has just begun. Many of these APIs expose services provided by legacy software. As Ex Libris completes development of its Universal Resource Management (URM) product, the possibility of a more comprehensive and consistent set of APIs may be realized.

We noted much more support for open APIs in Aleph 500 than in Voyager, which Ex Libris acquired from Endeavor Information Systems.

Ex Libris has the most at stake in this arena. The centrality of this issue to this company is reflected in the detailed response to our survey questions and in the number of customer sites interested in relating their experiences. While some of the other companies have some portion of their customers that may have an interest in making use of an API, Ex Libris focuses on libraries that consider these capabilities as a basic expectation. Large research libraries, national libraries, and consortia require systems that they can extend and expand. While a customer-facing API may be a minor consideration for other companies, it's a basic expectation for the kinds of libraries that Ex Libris aims to serve.

SirsiDynix offers one of the most comprehensive APIs for its flagship Symphony in the field of ILS providers. Sirsi Corporation, one of the antecedent companies of SirsiDynix, pioneered the territory of putting a full API into the hands of the libraries that use its product. The APIs made available at that time for Unicorn continue as that system evolved into the product now offered under the Symphony brand. That family of APIs, while comprehensive, operates through proprietary command-line utilities. Yet, with a library programmer trained in their use, they give access to every element of functionality and data. By policy, access to the API was originally limited to libraries that participated in the company's training program to mitigate the complex support calls related to its use. Today, SirsiDynix indicates it has relaxed these restrictions.

More recently, SirsiDynix has developed a growing set of APIs that follow a more open architecture. Though the number of APIs available through RESTful Web services is relatively small, it represents important progress in the evolution from a proprietary API to one more consistent with current technology preferences and that will be more easily consumed by its customer libraries. At this point, the number of Web services that it offers represents a small subset of the functionality of the whole system as seen in its proprietary API.

In recent years, **Innovative Interfaces** has increasingly focused on developing technologies that provide more open access to its products. Traditionally, Innovative has operated as a turnkey vendor that takes full responsibility for the products as they are used by its customer libraries and has enforced fairly tight control over the management of those systems. That approach feeds perceptions that the company's products are less open. The company has created a number of components that deliver APIs in specific functional areas. While the number of API products may be limited, the business model of licensing them discretely seems defensible given Innovative's general model for software pricing. In recent years, the company has been engaged in the development of Encore, which embraces a modern, service-oriented approach. The (separately licensed) Encore Query API provides a strong set of Web services that will be of benefit to libraries that require programmatic access to their systems.

The open source **Evergreen** ILS, the most recently minted product of the group represented in this report, may be the one that can most defensibly claim a service-oriented architecture. Its foundation on the OpenSRF framework forms the basis of a set of granular services upon which the higher application programming is built. The OpenSRF API seems at this point to be consumed mostly by developers of the system, and less by the libraries that have implemented Evergreen. Again, the demographics of the libraries that use the system come into play. Because the system is implemented primarily by consortia of public libraries, the requirements for extensibility by library customers through the API may be at a lower volume. But as Evergreen moves more into academic libraries, demand for this capability will likely increase. The recent implementation of Evergreen by the Conifer consortia of academic libraries in Ontario may pave the way for others. Today we see Evergreen in an early stage of its maturity cycle. Over time, we expect additional functionality built into the system itself and functionality offered through open APIs.

While **Koha** is an open source ILS, the profile of libraries using this software is not one where interoperability through APIs is a large concern. Most of the libraries using Koha have relatively modest automation scenarios where the automation system as delivered must meet their requirements. The libraries using Koha span a wide range of sizes and types, but the majority of them are relatively small. As a whole, these are not the kinds of libraries that face needs that would be addressed by a user-accessible API. While Koha includes some APIs, at

this stage they appear to be only minimally documented and used primarily by the programmers that work with the source code of the application.

Talis has established itself as one of the primary companies in the library automation industry that embraces SOA as the key strategic technology for libraries. The company has made a great deal of progress in creating software that helps libraries take existing legacy automation applications and create opportunities for interoperability through Web services. Just as important, the company has taken a leadership role in educating the library community on the benefits of SOA. Keystone, Jangle, and other products and projects have advanced the state of the art in this area of library technology.

Alto, the Talis LMS, does not offer a robust set of APIs itself, but rather enables this level of access to become available through Keystone. Keystone connects to Talis, or another LMS, through proprietary techniques, and is then able to expose a set of open APIs in the form of Web services. Libraries using Alto do not gain these points of interoperability unless they also license one or more of the Keystone modules.

At this time, Talis has not developed an entirely new library management system built using SOA. This suggests that Alto and its other legacy products will evolve over time to embrace this architecture. For the present, Talis' focus has been more on encapsulating legacy systems with integration layers that allow increased interoperability through Web services.

The Library Corporation takes different approaches in regards to APIs for each of its products. Library.Solution, given that it focuses on smaller libraries outside the bounds of those that require extensible systems, does not offer APIs directly. TLC points out that the systems are flexible and highly configurable in other ways, but not through this approach. Carl.X, which evolved from an older legacy, serves a larger class of libraries that come with the more advanced automation needs that involve access to APIs. TLC indicated that it has created and made available APIs as projects arise that require them. Carl.X includes a more comprehensive proprietary API, made available only to its customer libraries.

The new LS2 platform embodies a design that will be much more amenable to Web services and other APIs. TLC already positions the LS2 PAC, its initial module, as the means to deliver Web services to libraries using either Carl.X or Library.Solution as their ILS.

Polaris offers access to its product in several different ways. The most comprehensive level of access is accomplished through open access at the database level. The Polaris API offers its customers an option to work directly with the underlying database by providing them access to the full database schema. Like many of other the ILS vendors, Polaris licenses the API separately from the base product. This low-level database-oriented API gives libraries that use Polaris complete access to all aspects of the data managed within the system. Polaris offers far fewer higher-level APIs. A set of Web services delivered through SOAP provide read-only access to elements needed for online catalog displays such as location, call number, and current circulation status.

For **VTLS**, a company that specializes in customized products for complex library automation environments, APIs play an important role. The company offers different approaches to APIs across its product line. VTLS did not provide detailed information regarding the APIs implemented in its Virtua ILS, but the responses indicate that the API addresses some, but not all functionality and data managed within the Virtua application. Its new Chamo discovery interface embodies an API oriented more toward the underlying database than to the higher-level Virtua application server.

Conclusions

The evidence gathered in this report reflects ongoing progress toward more openness in library automation systems, but also that much work remains. We see a variety of options and opportunities. Libraries that expect to work with their automation system as delivered and not become involved in local extensions or programming will find that the majority of systems were built for that kind of use. For libraries that want to do more with their automation systems, however, we see a great deal of functionality possible today through open interfaces with momentum toward creating much more.

The hype persists. Library automation systems, proprietary and open source alike, compete more and more on the basis of enabling libraries to do more with their systems. That competition for openness drives the development of the technologies that enable that capability. The reality is still a bit messy. While we've seen a great deal of functionality exposed through Web services and other APIs, it still takes a lot of hard work to use the APIs in ways that benefit the library. The APIs available to library programmers continue to be quirky and less than comprehensive, even from the vendors with the strongest offerings in this area. We can also tell by the information received that vendors and libraries alike see the need to make systems more open. Hopefully, a better reality will evolve over time.