Software as a Service Comes to CAL

Executive Summary

By integrating an outside vendor's Software as a Service (SaaS), CAL's business functions were brought up to par quickly and efficiently enabling us to comply with student privacy regulations, eliminate error-prone duplication of data, add data security, and greatly improve the Office of Student Development's case management software experience.

Using software from outside vendors presents its own set of problems, many of which are not trivial, but we overcame these challenges. This white paper discusses the issues and methods used to surmount these issues.

The Business Problem

When the Office of Student Development (OSD) needed to replace Student Conduct Case Management software, they surveyed the marketplace for Software as a Service (SaaS) vendors and settled on Symplicity, Inc. The company's Student Conduct product, Advocate, had desired features that OSCAR, the case management software custom-created by IST, did not. Importantly, Advocate would enable case managers to comply with the complicated new set of federal regulations from the Family Education Rights and Privacy Act (FERPA).

The Technological Problem

However, every business solution generates new technological problems. In this case, OSD needed to get Symplicity its student data in real time, the transport of this data needed to be secure, and the students' privacy, protected.

Also, on the OSD side, there was the issue of extracting the right data: for instance, which classes students were taking, and clubs they joined. Answering these questions and collecting data from multiple places presented challenges to be overcome.

Symplicity offered two methods of data synchronization---online integration via web services, or uploading a batch file. Formerly, IST would have done a data load from the Bear Facts Student Information System. However, the new FERPA regulations prevented handing over the entire student data profile information when only one student's information was needed. Data needed delivery to Symplicity strictly on an as-needed basis.

Also, IST did not want to build any more systems around data loads. One problem with data loads is that batch files are typically a day behind thus exposing the data to errors and inconsistencies. Another problem with the old method was that custom solutions result in tight coupling that may not survive substantial future changes to the data source.

Most of the student information that Symplicity needed lives in the Bear Facts Student Information System (SIS) along with some data located in the Club Affiliation and Athletics databases. A primary objective of this integration effort was to enable campus software systems to reuse each other's data and functionality easily and cost-effectively.

The API and Integration Solution

To be responsive to the business needs of the university, IST rapidly deployed this package. To quickly and cost-effectively integrate the new SaaS with CAL's legacy system we gave Symplicity a simple web service. Symplicity only needed to provide a student ID to obtain just the necessary student data on a

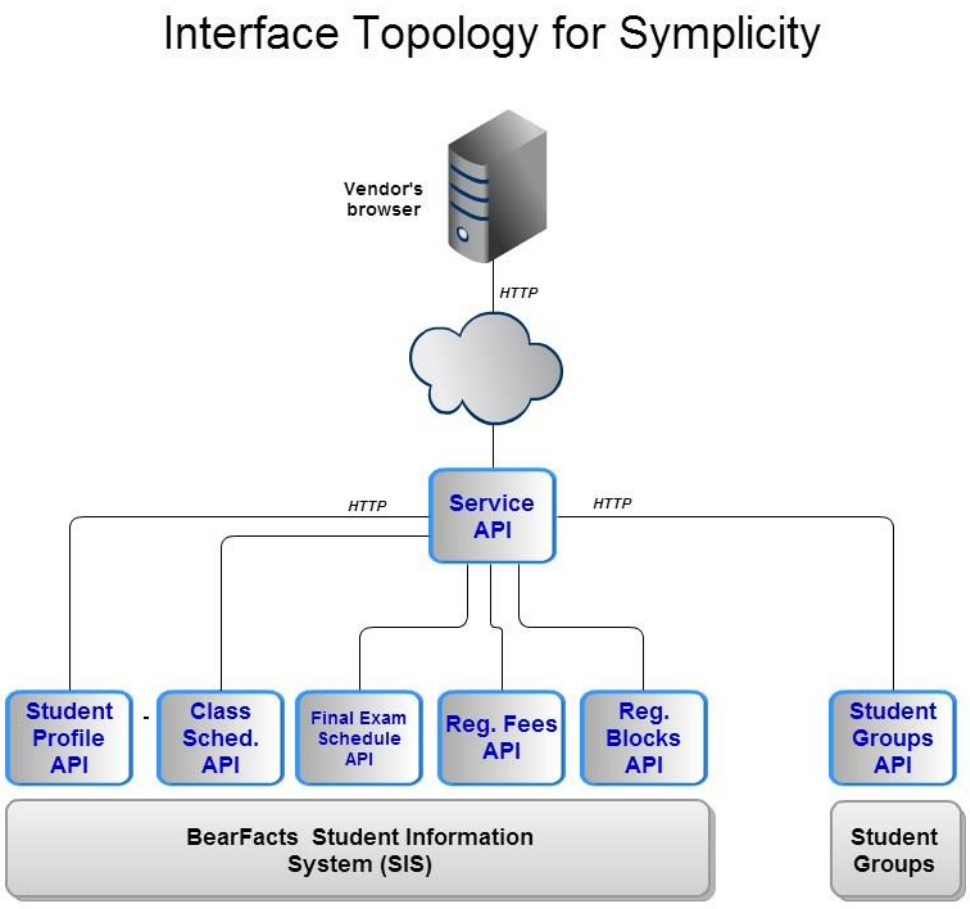
case-by-case basis.

We built end-to-end integration by creating a Service API that interacted with Symplicity, added System APIs to two legacy campus systems, using the Service API created in the first step to pull, aggregate, and format data from the System APIs. Finally, we forwarded the data to Symplicity's web browser

Access to CAL's internal systems, collecting, and aggregating the data were all handled on our side with the details hidden from Symplicity.

Once we created the web service, we secured the web service with a token-based system to handle authentication and authorization, and encrypted traffic by enforcing SSL on ServiceMix (the container for the Service API) and Tomcat, the server for the System APIs.

The diagram below shows the flow of requests between Symplicity and two API layers, the System APIs which pull student data from our BearFacts Student Information System, and the Service API that aggregates and formats the data for Symplicity.



By creating a Service API that coordinated, integrated, and processed data coming from the BearFacts and SGA APIs, this service, known by many names including, a 'Business,' 'Mashup' or 'Facade API,' hides the details of the System APIs from Symplicity. We used CXF as the web service engine that runs in Servicemix, an OSGi container.

We wrote code to create a JAX-RS endpoint and package it as a BUNDLE. (Note: the Felix plugin may be required to generate the necessary META-INF file.) Then deployed the BUNDLE to Servicemix and used a browser to view the new API.

See Adding JAX-RS endpoints to ServiceMix for an elaboration of these steps and code snippets.

Finally, we used tokens for authentication and IP verification, which exists at the server level to secure the physical and data transport layers.

In Conclusion

A services-based integration solution proved to be a powerful alternative to the pre-existing approach. IST built an integration component that handled the complexities of accessing one or more related data sources, then allowed access to that component through an easily consumable service by widely supported HTTP and RESTful APIs.

This approach eliminated error-prone duplication of data, improved security, and improved OSD's case management software experience. The Office of Student Development now enjoys a robust, thoroughly tested software experience that complies with FERPA that can now be regularly updated.

See Also

For the technical discussion of this case study, see Customizing Consumers' Experiences.