Consortial-Based Customizations for New Primo UI

Users interested in customizing their Primo installation are required to configure specific settings, files, and code during the View setup process. A consequence of this is that unique customizations are not easily sharable between institutions. With the release of the new Primo User Interface, Ex Libris has enabled institutions to manage interface customizations via the Package Customization Manager. In the summer of 2016, an Orbis Cascade Alliance working group investigated the efficacy of the Package Manager as a means of centrally sharing and deploying Orbis Cascade Alliance Primo Toolkit customizations. By virtue of passively loading customizations to the central package, each institution could pass custom parameters with local JS in order to adapt central customizations to the specific needs of that institution's users. This article will address both the potential and the limitations of the Primo Package Customization Manager. It will also provide best practices for consortia seeking to centrally manage and share Primo enhancements and it will identify areas of future development for centrally shared customizations.

by Dan Moore and Nathan Mealey

Introduction

The Orbis Cascade Alliance is a network of 39 resource sharing institutions in Washington, Oregon, and Idaho that migrated to a shared ILS Alma/Primo installation over multiple years from 2013 to 2016. Due to the nature of the shared ILS, it became necessary to establish teams and working groups to address particular issues, identify areas for development, and experiment with configurations in various components of both Alma and Primo. Each institution is granted their own Primo instance that they may customize to fit the needs of their local users.

Users interested in customizing their Primo 4 installation are required to configure specific settings, files, and code during the View setup process, sometimes requiring users to upload individual CSS files, HTML footer and header snippets, JavaScript, and additional content files independent of one another. A consequence is that customizations are not easily sharable between institutions; similarly, institutions using multiple Views of Primo for different libraries are required to manually implement each customization into each View of Primo. Further, customizing the interface requires a relatively high degree of familiarity with Primo architecture, markup languages like HTML, and programming languages like JavaScript.

With the release of the new Primo User Interface in August 2016[1], Ex Libris has enabled institutions to manage the new AngularJS-based Primo UI via the Package Customization Manager [2]. Additionally, the Package Customization Manager allows consortia to share customizations via a centrally configured package. Using the Package Customization Manager, users are able to organize their customizations with a locally managed repository uploaded as a .zip archive that may then inherit and build off a consortial Central Package.

In the summer of 2016, the Primo Consortial Configuration Working Group (PCCWG), an Orbis Cascade Alliance working group consisting of four individuals from different Alliance member institutions, investigated the efficacy of the Package Customization Manager as a means of centrally sharing and deploying Orbis Cascade Alliance Primo Toolkit customizations. By loading customizations to the central package, each institution could pass custom parameters with local JavaScript in order to adapt central customizations to the specific needs of that institution’s users. This article will address both the potential and the limitations of the Package Customization Manager. It will also provide best practices for consortia seeking to centrally manage and share Primo enhancements and will identify areas of future development for centrally shared customizations.

Background

In 2015, the Orbis Cascade Alliance initiated a project to develop a suite of shared customizations for the Primo 4 user interface. Motivated by a desire to take advantage of work that had already been completed and tested by several individual member libraries to customize their Primo instances, the project’s goal had been to determine how best to share these customizations so that libraries throughout the consortium could selectively take advantage of them.

The principal challenge with this effort was the practical reality that different libraries within the consortium have different levels of technology skills available among their staff. While some consortium libraries have a number of technology staff and several developers, others have no dedicated technology staff at all. Thus, the solution the project arrived at needed to accommodate and overcome this challenge as best as possible.

The resulting solution, the Primo Toolkit, offered a curated, browseable, and searchable inventory of customizations for the Primo 4 interface that had been developed by Alliance libraries. The items included had been prioritized by staff at member libraries, and so targeted those functionalities of interest to the greatest number of libraries.

Overall, the Primo Toolkit sought to accomplish the goal of sharing a range of Primo customizations such that as many libraries could take advantage of them as possible, by ensuring that the documentation and approaches used for the shared items was as consistent as possible. Terminology, included elements, and consistent references were standardized across items, and a great deal of effort was put into providing
step-by-step instructions that were as clear as possible. Ultimately, the hope was that even a novice programmer or website admin could implement the majority of the customizations.

Relative to the possibilities raised by the customization package manager in the new Primo UI, the approach embodied in the Primo Toolkit is very limited. But the project laid the groundwork for the Primo Consortial Configuration Working Group by creating a curated list of customizations to target, and firmly establishing the goal of sharing customizations across Alliance libraries as effectively as possible.

Technology

![UI customization Package Manager](image)

The Customization Package Manager allows users to manage a .zip archive of CSS, HTML, JavaScript, and image files that are loaded into the new Primo UI and override default Ex Libris styling. It is accessed via the Primo Back Office (above figure). Prior to the New Primo UI, customizing Primo required users to upload individual HTML and CSS files that were then selected during the View setup process. With the New Primo UI, customizations are collected into one archive that can be more rapidly deployed and shared.

Each institutional package is associated with a View. Ex Libris provides users with a Template Package that contains empty files in which users may add new customizations. This Package may act as a model for the View’s package or can be used to restore a View to default configuration. Upon unzipping the archive, the user will discover a root folder that is by default named VIEW_CODE. It is imperative that the name of this folder be identical to the View to be customized. In the example above, the root folder of the archive would need to be named “OSU” in order for the system to accept the file. Live validation is present in the system so an uploaded package with an incompatible view code will be rejected. Users zipping the archive on a Mac machine would also need to confirm that any .DS_Store metadata files are not present; at the time of this experiment, the Primo Back Office automatically rejects any .zip archive with this file in it.

At the time of our testing, packages were organized in the following manner:
The organization of the package is hierarchical and intuitive. All CSS rules are stored in custom1.css, all edits to the landing page of the new Primo UI are stored in home_en_US.html (although header navigational items are still controlled by the Main Menu Tile in the Primo View Configuration Wizard), custom images are stored in the img folder, and JavaScript is stored in custom.js. This structure is identical between Central and institutional packages.

On the Customization Package Manager screen, users configuring packages and uploading for deployment may toggle “Inherit from Central package” to load the Central package into the selected View. Participation in the Central package is voluntary and is configured on the specific View. This allows greater freedom for individual institutions to have separate views that might not require the Central package for local needs or for local testing. For example, if an institution uses shallow client browsing terminals that would not benefit from the heavy customization meant to address users who are logging longer search sessions, it may select not to inherit customizations from the Central package.

At the time of our testing, additional files in any folder except the img directory were not actionable by any means in the new Primo UI. The group tested with additional HTML and CSS files and were not able to call or reference those files during page load. As this is an evolving feature of the new Primo UI, which itself will undergo another major release update in November 2016, it may be possible in the future to host additional CSS, HTML, or JavaScript files in the package.

Recommendations for consortial partners

The Package Customization Manager, both local and central packages, offer great value to consortia seeking to share enhancements; additionally, individual institutions seeking to take advantage of consortially loaded customizations will benefit from reduced development and implementation time. Despite the low early adoption rate of the new Primo UI in Alliance libraries, we have identified multiple use cases for the central package. These include: deploying material type icons centrally so that resource type graphics are unified across the Alliance; publishing CSS styling to make each institution’s resource sharing request forms similar; loading JavaScript triggers for hiding unneeded actions or adding new ones based on popular Alliance resource sharing workflows. At the time of our testing, however, we discovered that any customizations made to the HTML template in the institutional package would entirely override any customizations made to the HTML template in the central package. It is not feasible to use this technology in its present state to deploy static HTML customizations.

The Central Package may be used to deploy a consistent set of images throughout the consortia as a means of standardizing the display of common resource types as well as other common Primo features. We foresee several situations where this will be a virtue. With resource icons standardized, screenshots of search results may be shared openly throughout the consortia; barring significant style choices that might make one institution vastly different from another (such as a different color scheme or font styling), it may be helpful to crowd-source asynchronous training modules that can then be deployed to both local and distance users of the discovery layer.

A standard set of material type images will support the Alliance’s resource sharing program by reducing barriers for users at any consortial library. Furthermore, standardizing resource type icons with a consistent set of images that are curated by the consortial partners itself means future changes to image sets published by Ex Libris may be safely disregarded. It is important to note that the /img directory within both the Central and institutional packages can act as a file store for custom images used throughout the Primo interface. Early experiments revealed this to be a strong candidate for hosting infographics or other media used within the discovery layer; however, due to the upload times of the central package and timelines set forth for the experiment, the authors did not test the limits of either file size or file types stored in this directory.

http://journal.code4lib.org/articles/11948
Due to the nature of and need for individual styling at member institutions, we foresee the value in shared CSS customizations to be less than the value of shared images or custom JavaScript. However, there is still value, based on resource sharing workflows within consortia, in using the centrally loaded style guide to standardize the display of request forms across all member institutions. The central package is an ideal candidate to deploy any style fixes to the default Primo layout that may boost accessibility. It is important to note that the institution package CSS file is loaded last and will override both style rules in the native new UI as well as rules in the Central Package. This affords individual institutions greater freedom to break away from the pack, but we recommend this only if the users of that institution will benefit more from those unique customizations than adhering to consortial styling.

Centrally shared JavaScript offers exciting promise for the future of the new Primo UI. Early experiments within the working group have demonstrated that JavaScript functions stored within the central package can be triggered, modified, and customized by JavaScript code stored in the institutional package. An early experiment conducted by the Working Group began by uploading a basic JavaScript code snippet to the central package's `custom.js` file:

```javascript
function writeToConsole(p1) {
    console.log("triggered from central package by "+p1);
}
```

In the local package's `custom.js` file, the function was called with additional parameters:

```javascript
writeToConsole("OSU");
```

Upon page load, the console records the correct log declaration:

```
triggered from central package by OSU
```

**Figure 3:** Console log entry.

One promise of this feature is that it is plausible for the central package to act as a code repository for consortium-wide JavaScript customizations. Individual institutions can modify their institutional package to enable those customizations and configure how those customizations work. For example, one popular JavaScript enhancement shared through the Orbis Cascade Alliance Primo Toolkit is adding a “Report a Problem” link to the Actions Menu on the brief record and full record pages. Clicking this link redirects the user to a locally-configured Drupal (or other CMS) form. Within the new Primo UI and taking advantage of the central package, the JavaScript code for this enhancement can be loaded and shared natively with all members. Members can then modify the central code by supplying different parameters that will redirect users to the appropriate forms. Furthermore, additional code in the local package might enable that institution to gather specific data from other Primo data modules. Further experiments are required to determine the efficacy of using the Package Customization Manager to centrally deploy JavaScript code for this purpose but early results are promising.

In order to make the most of this technology, it would behoove consortia to develop and organize task forces charged with directing development, managing customizations, and training users. Specifically, the PCCWG identified three primary goals for the fall and winter of 2016: develop training modules to initiate Alliance users into AngularJS, codify best practices for creating and publishing customizations via the Central Package, and investigate a greater extent of use cases to explore. Depending on the current status of a consortium's shared implementation of Primo, it may be advantageous to commit to developing a central package as part of the consortial agreement.

**Conclusion**

Adopting the Customization Package Manager as the vehicle for sharing customizations within the Alliance is a promising venture. We have identified a few areas that require special consideration before widespread adoption can take place. When transitioning from developing for Primo 4 to the new Primo UI, a primary hurdle is adapting to the AngularJS framework. While this will impact future development of new enhancement, it also requires a review of existing customizations published in the Primo Toolkit. Users who are already live with Primo and have customized their UI extensively should pay particular attention to the compatibility of jQuery and AngularJS; it was discovered in our experiments that several popular jQuery-based customizations were not compatible with the new Primo UI and would have to be rewritten.

As discussed above, the virtue of distributed development of customizations is an attractive quality of the Package Customization Manager. Developing across the consortium in this manner saves precious development time and resources by avoiding ‘reinventing the wheel.’ This also allows the consortium to share with its smaller institutions, or institutions that lack a dedicated development team or technology specialists, attractive customizations that enrich the Primo experience.

It would benefit consortia to create internal training modules to acquaint library programmers and Primo users to AngularJS. While a significant portion of a Primo View are still customizable with the Views Configuration Wizard in the Primo Back Office in the new Primo UI, the entire front-end has been built using AngularJS and makes use of its directives to display information. Librarians and programmers accustomed to the current Primo 4 interface will need to understand how AngularJS works at a fundamental level in order to best write customizations for the new interface. Additional experiments are necessary to confirm compatibility and best practices with current widgets and external resources that some members of the global Primo community have programmed into their Primo 4 UI.

The new Primo UI does not supplant or replace the existing Primo 4 UI. This concurrent release and support on Ex Libris' behalf allows institutions individually and consortia as a group to decide when to go live with the new user interface. Considering the significant changes the
new user interface present as well as the new technologies that underlie it, it may benefit institutions and consortia to maintain the existing Primo 4 UI while current development on the new UI is underway. Ultimately, the Primo Consortial Configuration Working Group is optimistic that the Primo Customization Package Manager will enrich the shared Primo experience throughout the Orbis Cascade Alliance.

Notes

[1] Early discussion of the new Primo UI referred to the new user interface as “Primo 5.” Ex Libris has stressed that the new user interface is a separate release and the authors will maintain the separation between the new Primo UI and Primo 5 in this article.

[2] Documentation may be found here: https://knowledge.exlibrisgroup.com/Primo/Product_Documentation/Back_Office_Guide/090Primo_Utilsities/The_UI_Customization_Package_Manager


About the Authors

Dan Moore is Discovery Services Librarian at Oregon State University, dan.moore@oregonstate.edu.

Nathan Mealey is Manager of Library Technologies at Portland State University, mealey@pdx.edu.