Primo Consortial Configuration Working Group: Report & Recommendations

Overview

The Primo Consortial Configuration Working Group (PCCWG) was formed by the Systems Team in spring 2016 to explore the consortial functionality built into the new Primo User Interface (UI) and provide recommendations for:

1. What customizations to consider sharing centrally (CSS, HTML, JS, etc.)
2. A description of what the impact would be for member libraries (workflow, Primo improvements, etc.)
3. A workflow for sharing and maintaining them (describing impact on Alliance central and member staff)

The members of the PCCWG included:

- Nathan Mealey, Portland State University, Chair
- Holli Kubly, University of Oregon
- Dan Moore, Oregon State University
- Jeremy McWilliams, Lewis & Clark

Overview

The PCCWG investigated this new functionality within the limitations of the Primo May release, which was not a production-ready version of the new UI. While this limitation prevented the group from being able to thoroughly understand some specific functionalities, on the whole we were able to come away with a clear sense of how this functionality will work when the new Primo UI is production-ready. In brief, it offers the Alliance a significant opportunity to share Primo interface customizations across Alliance libraries in a way that enables individual institutions to opt-in to specific customizations while also significantly minimizing the barrier to these customizations being adopted locally.

Summary of consortial functionality in the new Primo UI

The consortial functionality included in the new Primo UI enables consortiums that have both a centrally-managed Primo Back Office (PBO) and separately-managed institutional PBOs to create and share customization packages. These packages include HTML, CSS, Javascript, and images, and can also be created and loaded to institutional PBOs by local administrators. When a centrally-shared package is available, institutions may elect to opt-in and inherit from this package. Without opting in, any elements included in the centrally-shared package will not affect an institutional PBO.

The new Primo UI employs a customization hierarchy, where centrally-shared packages can override elements in the out-of-the-box configuration, and locally-created customizations can override elements in both the out-of-the-box configuration and the centrally-shared customization package. As a result of this hierarchy, customizations can be created and shared via the central package in such a way that institutions can inherit from the central package while still selectively taking advantage of specific elements that it includes.

For example, if an image file named “icon_article.png” is included in the central package, this would replace the default article icon in Primo for all institutions that have elected to inherit from the central package. At the same time, an institution that has opted to inherit from the central package can include their own file named “icon_article.png” in their local customization package, and in doing so will override the centrally shared image (and the out-of-the-box image).

A more powerful example of how this hierarchy can be employed would be centrally shared Javascript customizations. For example, if the Alliance opted to centrally share a version of the Primo Toolkit Javascript customization that adds a facet in Primo to send the user’s search term(s) to an external system (such as Worldcat or Google Scholar). In this case, all of the Javascript code needed to make this functionality work in an institutional Primo instance would be added to the central package, and configured in such a way that an institution would be able to trigger it in order for it take effect. To do this, an institution that wanted to take advantage of this functionality in their local Primo instance would add a small Javascript snippet to their local customization package that would trigger the centrally-shared Javascript code, and would add the facet to their Primo instance.

There are two important takeaways from this example:

1. Taking advantage of Javascript functionality that is shared functionality would remain optional for Alliance libraries, even if they have opted to inherit from the central package.
2. Implementing the Javascript snippet needed to enable the centrally shared Javascript functionality would require minimal skills, and would be within the skill of all libraries in the Alliance (particularly with support from the Systems Team).

Lastly, this approach could be applied to a wide range of potential customizations, and would enable the Alliance to take significant steps forward toward elevating the overall level of functionality in institutional Primo instances. Most importantly, this functionality enables this to be done in a way that minimizes the amount of local expertise that would be needed.

Implications of this functionality

The Primo Toolkit was the Alliance’s first effort to develop and share customizations in a coordinated way for all member libraries to take advantage of. The consortial functionality in the new Primo UI is a more advanced means of accomplishing this, and would enable the Alliance to make the type of customizations included in the Primo Toolkit more readily available for member libraries to implement.

At the same time, there are some considerations to take into account:

1. In theory, most customizations developed by member library staff could be considered for inclusion in this framework. In practice, there will need to be some sort of Alliance-organized workflow for curating what items are included in the central package, how they are incorporated, and how they are maintained. The Systems Team or a

dedicated sub-group of the Systems Team are perhaps the optimal starting points for building out this workflow.

2. The impact on central Alliance staff is potentially minimal, depending on how the workflow for curating included customizations is devised. Central Alliance staff time will be needed for loading and deploying changes to the central package. But this work requires only a small amount of dedicated time. If the Alliance implements a sound workflow for curating the included customizations, much of the work can be done by Team or sub-group members.
3. It will be important for the Alliance to prioritize included customizations, and approach use of this customization framework with caution at the outset. There is likely a tipping point beyond which including additional customizations makes ongoing maintenance of them difficult to sustain. If/when a workflow is developed for curating these customizations, addressing this concern should be one of the workflow's ongoing priorities.

Recommendations for moving forward

Below are specific recommendations for how the Alliance should move forward from here to take advantage of this functionality in the new Primo UI.

1. The Alliance should organize a working group to use what the PCCWG has learned to pursue 3 goals:
    a. Identify a range of images-based, CSS, and Javascript customizations to implement as pilot tests, based on known customizations that are in place or have been identified as needs.
    b. Implement the list developed via the above goal.
    c. Provide recommendations for an ongoing workflow for developing and maintaining customizations included in the central package, and for managing the updating and deployment of changes.
2. The recommendations of the above working group should be followed up on by Alliance staff, in conjunction with Team chairs as needed, developing and putting in place workflows to manage these ongoing processes.
3. The Alliance should create a working group to provide recommendations on how best to create a forum for ongoing customization of the new Primo UI within the Alliance. Potential goals include:
    a. Building AngularJS skills within the Alliance
    b. Developing best practices for writing and sharing customizations via the centrally-shared customization package and/or Primo Toolkit
    c. Identify and implement specific functionality to develop as shared solutions