Definition of the Okapi-Stripes Platform, FOLIO LSP Base and FOLIO LSP Extended Apps

22-Mar-2019

- Platform Description
- Okapi-Stripes Platform Components
  - Runtime components and frameworks
  - Fundamental application building blocks
  - Software Development Lifecycle Practices and Utilities
- FOLIO Library Services Platform (LSP)
  - FOLIO LSP Base
  - FOLIO LSP Extended Apps
  - Implications of Platform vs LSP Base vs Extended Apps

Platform Description

The Okapi-Stripes Platform – consisting of Okapi, Stripes, and related components – was initially developed to be the infrastructure for the FOLIO LSP project. It is seen now as a generally useful infrastructure component for other projects. The most well-known project to use the Okapi-Stripes Platform is the “FOLIO Library Services Platform (LSP)” suite of apps intended to replace the functionality of an integrated library system. The Okapi-Stripes Platform, however, is problem-domain agnostic: it enables developers to build software solutions that solve many problems in the library space. (In fact, it would facilitate solutions even outside the library domain.)

The Okapi-Stripes Platform is a true multi-tenant application environment for web-based applications (or "apps"). The concept is similar to Google’s office suite and other web-hosted, extensible environments. A tenant is like a workspace in the Okapi-Stripes Platform, which is typically specific to a library or organization. It has its own set of apps, its own users, and its own data. Multiple tenants, each using a variety of apps, run on a single installed instance of the Okapi-Stripes Platform. Apps are deployed independently for each tenant, so libraries can adapt an Okapi-Stripes Platform-based solution to their needs by selecting which apps to use.

One of the Okapi-Stripes Platform's critical design decisions is the division of functionality into Apps. Apps correspond roughly to business objects in the supported domain. For instance, one would have an app for Invoices, another app for Vendors, and yet a third app for Payments. Each app is responsible for the business logic and data storage for its type of data, and other apps communicate through well-defined, versioned application programming interfaces (APIs). Between tenants, apps also communicate through these same APIs. To have an app that reads data directly from another app or another tenant violates a fundamental separation of concerns built into the Okapi-Stripes Platform.

The Okapi-Stripes Platform establishes a basic software architecture and delivers foundational services that any application set would require. These services are general, utilitarian, and domain agnostic. More specifically, they do not assume, enable or support specific workflows within any domain. They are building blocks. In some ways, the analogy of an Operating System is appropriate for the Okapi-Stripes Platform. In and of themselves, the platform pieces are of little functional value. The domain-specific apps are required before organizations see the benefit of adopting Okapi-Stripes Platform. Once the Okapi-Stripes Platform is adopted, though, the incremental costs of creating or adding new apps are significantly reduced.

Okapi-Stripes Platform Components

The platform has three general pieces:

Runtime components and frameworks

The Okapi-Stripes Platform back-end consists of a microservices architecture called Okapi. Okapi is an implementation of the microservices API Gateway pattern. Modules are written to the Okapi specification to perform the business logic and data storage activities of an app.   The Okapi specifications insulate module implementers from needing to understand operational details of service discovery, load balancing, reporting, and tenant-separation. To aid in the creation and maintenance of Okapi modules, developers can use a framework called RAML Module Builder (RMB). Geared towards Java developers, RMB provides facilities for building JSON APIs and seamlessly storing JSON objects in a Postgres database as well as providing stub interfaces for the required Okapi lifecycle management functions.

A second part of the Okapi-Stripes Platform is Stripes: the front-end development toolkit and a set of reusable user interface (UI) components. Stripes allows teams to build user interface modules that nest within a single-page web application (SPA, Wikipedia definition). Stripes itself is built on the open source React JavaScript library, and it leverages the responsive design and internationalization support build into React. Stripes also facilitates the communication with back-end business logic and data storage services provided by Okapi modules.

It is possible for software applications to use other front-end development toolkits other than Stripes to communicate with Okapi modules. For instance, a library discovery layer tool may communicate directly with Okapi modules to look up items the patron has checked out. Most apps on the Okapi-Stripes Platform are expected to use the Stripes toolkit to provide a consistent user experience to app users.

Fundamental application building blocks

Regardless of the problem domain, every system requires users, permissions, authentication, configuration, and secure inter-communication mechanisms. The Okapi-Stripes Platform provides this functionality through domain-agnostic modules:

- mod-users: management of details of users in the system

- mod-permissions: permission-checking functionality that determines if a requested operation is allowed by the user
- mod-authtoken: session management
- mod-configuration: configuration management

On the roadmap for development is workflow functionality that will be available to app developers for managing to-do lists and orchestrating machine operations.

Software Development Lifecycle Practices and Utilities

The Okapi-Stripes Platform uses modern software development lifecycle practices such as a continuous-integration/continuous-delivery pipeline, automated test suites, standards around tooling, code coverage expectations, deployment patterns and established environments that allow development teams to integrate new code and functionality. Those practices are provided in the form of infrastructure management code (folio-infrastructure, folio-ansible, folio-install) and documentation.

Likewise, there are command line tools that are designed to allow system operators and developers to leverage the platform capabilities in a quick, convenient and automatable fashion. Those tools are the Okapi-CLI and Stripes-CLI.

FOLIO Library Services Platform (LSP)

This document is provided to help the community with terminology and concepts that can help define the different aspects of FOLIO and the software systems that are part of it. We expect that future projects will be modeled after FOLIO LSP and share the Okapi-Stripes Platform, but will have separate domain "Base" modules and "Extended" apps.

FOLIO LSP Base

To provide library-specific functionality on the level of contemporary Library Services Platform, FOLIO project is actively developing a set of backend modules and staff user applications that will eventually form a consistent solution to everyday tasks of running a library. Those modules are designed to run on the Okapi-Stripes Platform and are built both by the FOLIO Core Team and External Teams that closely follow FOLIO style and design guidelines. Those modules are fully replaceable and selectable, however, due to interdependencies between them it may be required to select full subsets of those apps to provide required functionality. We use the term FOLIO LSP Base to denote them. LSP stands for Library Services Platform.

FOLIO LSP Base includes, but is not limited to the following:

User Management, Inventory and Circulation suite of modules:

- mod-circulation
- mod-inventory
- ui-checkin/checkout
- ui-users
- ui-inventory
- ui-calendar

Acquisitions suite of modules: (non-exhaustive):

- ui-orders
- ui-vendors

ERM’s suite of modules

FOLIO LSP Extended Apps

FOLIO LSP Extended Apps are library-domain specific modules and apps that are provided by FOLIO partners. The key characteristic of “Extended Apps” is that their use is ancillary and optional in terms of running a library. Clearly they can be useful, but they are not essential. It’s generally encouraged that all FOLIO apps follow the common set of usability and look/feel guidelines, although this is not a requirement for extended apps. Note that there are no functional dependencies between FOLIO LSP Base apps and Extended Apps, as they operate side-by-side. FOLIO LSP Extended Apps are fully replaceable and selectable.

Examples of potential FOLIO LSP Extended Apps:

- Collection analysis tools
- Reporting tools for tracking physical conservation and digital preservation activities
- Integration with learning management systems
- Curatorial value assessment records and associated documentation
- Records of movement of items within an organization (typically used for museum collections)

Implications of Platform vs LSP Base vs Extended Apps

The distinction between these categories implies a certain level of “centralness” and with that comes added governance. The table below begins to outline the current thinking. Note that the table below is meant to convey directional intent. We currently do not have policies or guidelines that dictate this governance.

	Okapi-Stripes Platform	**LSP Base**	**Extended**
Development team	Core	Coordinated by Product Council	open
Technology stack	Approved by Tech Council	Approved by Tech Council	Team specific
Adherence to Definition-of-Done	Strict	Strict	Encouraged
Manage Pull Request	Core	Module-moderator	open