Folio Architectural Blueprint

WolfCon 2020

What is the Folio Architectural Blueprint?

The Folio Architectural Blueprint is the set of identified **strategic** architectural changes required for the Folio platform.

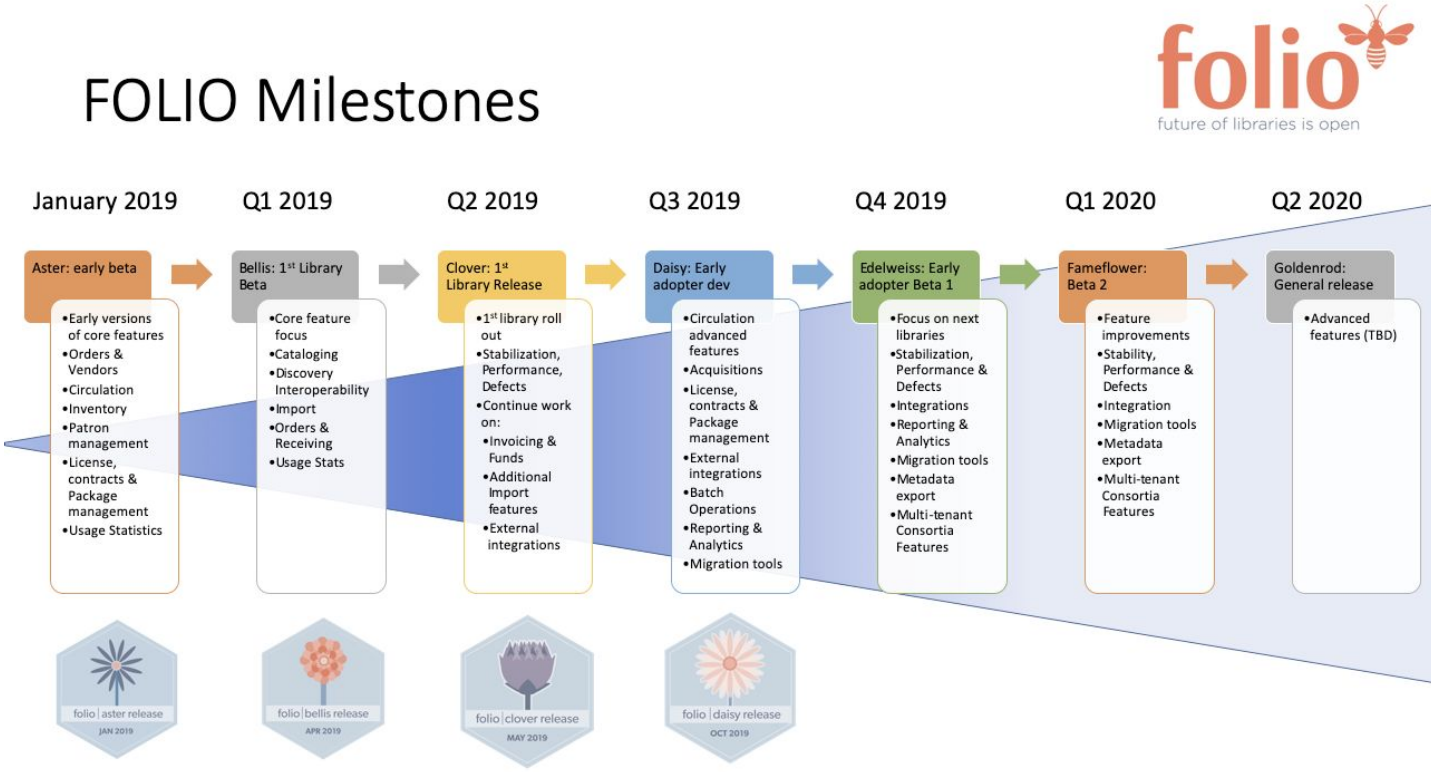
While distinct from the Feature Roadmap, some of changes may be required in **anticipation** of roadmap features.

In contrast to the Folio Tech Debt list, these reflect development in **new** areas of the platform, rather than the reworking of already competed areas.

What are the goals of this meeting?

- Present some candidate items to being added to a Folio Architecture Blueprint
  - Briefly present and discuss each item
  - Goal is NOT to design the solutions here
- Decide whether each item belongs on the Blueprint
- Roughly prioritize the timeframe of each item
  - “Now”, “Soon”, “Later” (or at most which Quarter to start)
- Be in a position to create a list of actionable items that can inform/affect:
  - Feature Roadmap
  - Tech Debt List
  - Development Schedule

Feature Roadmap



FOLIO Milestones

folio
future of libraries is open

January 2019 — Aster: early beta
- Early versions of core features
- Orders & Vendors
- Circulation
- Inventory
- Patron management
- License, contracts & Package management
- Usage Statistics

Q1 2019 — Bellis: 1st Library Beta
- Core feature focus
- Cataloging
- Discovery Interoperability
- Import
- Orders & Receiving
- Usage Stats

Q2 2019 — Clover: 1st Library Release
- 1st library roll out
- Stabilization, Performance, Defects
- Continue work on:
  - Invoicing & Funds
  - Additional Import features
  - External integrations

Q3 2019 — Daisy: Early adopter dev
- Circulation advanced features
- Acquisitions
- License, contracts & Package management
- External integrations
- Batch Operations
- Reporting & Analytics
- Migration tools

Q4 2019 — Edelweiss: Early adopter Beta 1
- Focus on next libraries
- Stabilization, Performance & Defects
- Integrations
- Reporting & Analytics
- Migration tools
- Metadata export
- Multi-tenant Consortia Features

Q1 2020 — Fameflower: Beta 2
- Feature improvements
- Stability, Performance & Defects
- Integration
- Migration tools
- Metadata export
- Multi-tenant Consortia Features

Q2 2020 — Goldenrod: General release
- Advanced features (TBD)

Architectural Blueprint Candidate Items

- Security: (fallout of the Security Audit)
- Refactoring Okapi
- Tenant Management
- Full Multi-Tenancy
- Adopt PubSub
- Support for GDPR
- Search Engine
- Users and Permissions
- Automation Engine
- GraphQL
- Database connectivity
- Codex
- Inter-Folio Integration

Security Audit Fallout

What?

The imminent security audit will formally identify security defects and vulnerabilities with Folio. A remediation plan will be put in place.

Addressing some of issues will require some architectural changes in Folio

Why?

A formal audit will elevate the visibility of known security shortcoming and identify new ones.

We have created a security policy for Folio and defined at a high level how to address security vulnerabilities. This will the opportunity to put those into practice.

Refactoring Okapi

What?

Okapi has evolved to assume too many responsibilities

- Proxying Gateway
- Tenant context (runtime)
- Dependency management (build)
- Registry (runtime)
- Tenant APIs (setup and upgrade)
- Various Features including; timers; post- and pre-filters; metrics

The multiples responsibilities would be split-out into their separate components.

Why?

Making changes in one area of Folio carries a high risk of introducing regressions in another area.

The current structure introduces security vulnerabilities. In addition to an increase surface area, some capabilities require the use of elevated permissions (e.g. Tenant APIs) while other capabilities must be publicly accessible (e.g. Proxying Gateway).

Allow for the independent evolution of the capabilities of each separated component.

Tenant Management

What?

Separate out the responsibilities of tenant management from Okapi.

Possibly two separate components:

- A administrative component for tenant provisioning and upgrading (including data upgrading)
- A runtime component for tenant registration and entitlements, (“mod-tenant”)

Why?

Improves the security profile by separating runtime capabilities from administrative capabilities

Provides the basis for Full Multi-Tenancy capabilities

Provides the basis for tenant management tooling (e.g. data migration)

Full Multi-Tenancy - (aka Consortial Support)

What?

Folio currently exists as a highly segregated multiple single-tenant environment. It is the goal that Folio should be a fully multi-tenant environment.

What is missing is cross-tenant capabilities.

This amounts to being able to split the existing tenant context into two: the originating tenant initiating operations and the target tenant on whose data to operate.

Why?

In preparation development of Folio Consortium features it will be necessary to support controlled and selective data sharing between tenants. The individual tenants who are part of a consortium need to be aware of each other and be able to selectively access each other's app data. (E.g. shared inventory, separate funds)

Full multi-tenancy with cross-tenant capabilities will allow larger institutions to chose to model themselves in Folio using multiple tenants

.

Adopt PubSub

What?

PubSub functionality has been developed as part of SRS. But is has been developed so as to be generally usable in the platform.

Identify pending features that would benefit from a PubSub solution and design those solutions prior to their development on the feature roadmap.

(Also consider applying PubSub to address technical debt items.)

Why?

Using PubSub for inter-app communication can greatly simplify the solution of problems requiring coordination between different apps. (E.g. Orders publishes a new order and Agreements only needs to subscribe to those.)

When applicable, using PubSub can produce a more robust system by reducing direct dependencies between modules and creating a loosely coupled system

Support for GDPR

What?

There is a need to provide support APIs to **facilitate** operations against Folio that originate with GDPR requests such as.

- The right of access
- The right to rectification
- The right to erasure
- The right to restrict processing
- The right to data portability

Adds to module responsibilities, to report user data

Why?

Fulfilling such requests for Folio currently would consist of crawling datastores. Individual scripts would need to be created and maintained.

Folio could provide APIs that automatically perform the desired actions across any relevant modules.

Search Engine

What?

The search capabilities of Folio have grown organically on top of the native search capabilities of PostgreSQL

Alternative approach is to use a dedicated search engine, such as ElasticSearch

Why?

As the scope of Folio changes, both in the number of components and in their complexity search performance begins to suffer.

There will come a point where no amount of turning of the search capabilities will be able to provide adequate performance across the entire platform. At that point a specialized search engine will be required.

User and Permissions

What?

Introduce tenant-level and system-level users.

Implement protections around those users to prevent their accidental modification through user management tools.

Introduce the concept of user Roles. Distinct from user groups these can have permissions directly assigned to them which individual users can automatically inherit at runtime.

Why?

Temporary workarounds are already in place to emulate tenant-level users or system-level users. But these are fragile.

There are security risks and performance issues inherent in a flat permission model

Automation Engine (aka “workflow engine”)

What?

Adopt an automation engine for use within Folio. The purpose of which is to provide systematic sequencing of predefined operations.

This introduces an abstraction layer in order to declaratively define workflows.

Why?

A declarative approach to workflows provides a more robust and flexible platform

May eventually be used to enable features such as Task Lists

GraphQL

What?

GraphQL is an alternative mechanism for requesting and retrieving data from Folio

- Consolidate multiple queries into one
- Taylor the data response to limit size
- Traverse the data models to simulate joins

Note that a mere convenience implementation to simply UI module development is not sufficient. This needs to address the performance issues inherent in the alternative approach: a business layer the merely calls underlying APIs

Why?

Long desired enhancement for Stripes to replace stripes-connect

Performance optimization to more effectively implement join operations through APIs

Database Connectivity

What?

RMB-based modules support database connectivity at the module-level only.

Extend that model to support database connectivity at

- Tenant-level
- Interface-level
- HTTP Method (GET, PUT, POST, ...)

Why?

This offers greater flexibility in how hosted systems may be hosted, particularly in a multi-tenant configuration. Possibilities include:

- Sensitive interfaces may use a different database for storage
- Specific tenants may be provided with dedicated storage
- Support read-only nodes with different optimizations from read-write nodes.

Codex

What?

Refresh and update Codex Search to fully support multiple data sources beyond the two existing.

Extend Codex Search to support more than just Instance searching, (E.g. packages)

Implement Codex features beyond searching. Such as relationship management.

Why?

Additional sources for Codex Search are becoming available

There is a need to search Codex for other objects such as packages. This can help reduce duplicative efforts involving plugins

InterFolio Integration

What?

Allow multiple Folio installations to work together. Distinct installations might provide specialized portions of Folio, yet appear as a single installation from the Tenant perspective

Requires a coordinated and distributed delegation between the multiple Okapi installations involved.

Why?

Allows for institutions to subscribe to a shared cloud-based Folio installation. But also be able to run a local Folio installation that appears to be part of same cloud-based Folio.

Allows local Folio installations to run local apps fully within their control. But rely on a shared hosted Folio installation for less critical Folio apps.

Fin