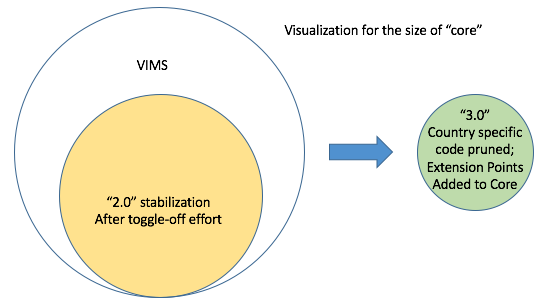
**Legacy OpenLMIS implementations have a migration path to new versions**

**Epic:** Legacy Migration Path

**Description:** The OpenLMIS Community commits to supporting countries with existing implementations in researching whether upgrading their implementation to a newer version of OpenLMIS makes sense for their situation. However, the community should understand that upgrading to OpenLMIS (post 3.0) will likely result in a decrease to the available feature set when compared to a country’s current implementation. The country or implementer driving the migration will be responsible for requesting extension points from the core team (as needed) and building any country specific functionality in one or more modules. As such, any decision to migrate should be thoroughly investigated. See the below table for activities the community can assist a prospective migrating party in undertaking.

|  |  |
| --- | --- |
| **Activity** | **Community will Assist** |
| Creation of a data migration script | Yes – via community forums |
| Creating core extension points | Yes |
| Creation of contributed modules (including functionality removed to achieve 3.0+) | No |
| Provide training on 3.0+ feature set | Yes – via community forums |
| Deploying 3.0+ code | No |
| Modifying reference UI to meet country requirements | No |



Some definitions:

**VIMS** = The last pull from the VIMS BitBucket to the OpenLMIS master branch [December 2015]

**OpenLMIS 2.0** = A reduced version of **VIMS** with country specific code, tables, and configuration toggled-off. New country implementations of OpenLMIS can be started from this branch. New implementations can take advantage of country specific code by toggling it back on. Starting with 2.0 [expected mid January 2016], code contribution standards will be enforced. Progress towards the 3.0 release tag will take place on the 2.0 codeline and current implementations are encouraged to contribute code (according to the contribution standards) and merge-in new releases as they become available. The first goal post-2.0 is to create a modular architecture. Once this “modularity” release occurs, the focus will shift to carving out core and removing non-core features (see **OpenLMIS 3.0**).

**OpenLMIS 3.0** = Time based release (November 2016). The following tasks will form the basis of the OpenLMIS re-architecture:

* Removal of country specific code (that had been toggled off to achieve 2.0)
* Creation of a tightly focused “Core” that contains global functionality in the LMIS domain; has high test-coverage, high reliability, and (eventually will) change relatively infrequently.
* Creation of extension points so that country specific behavior can be built in modules

**Data Migrations**

From **VIMS** to **2.0**, data migration should be trivial.

Since **3.0** is a release on the same codeline as **2.0**, no large scale data migration will be necessary for those implementations that are able to merge updates (although some changes to schema may be necessary when deploying updates from 2.0 to 3.0). For implementations that fork, the time commitment for a data migration to later versions (3.0 onwards) will vary depending on when they fork.

**Wish List:**

* Data migration script for migrating from **VIMS** to **2.0**