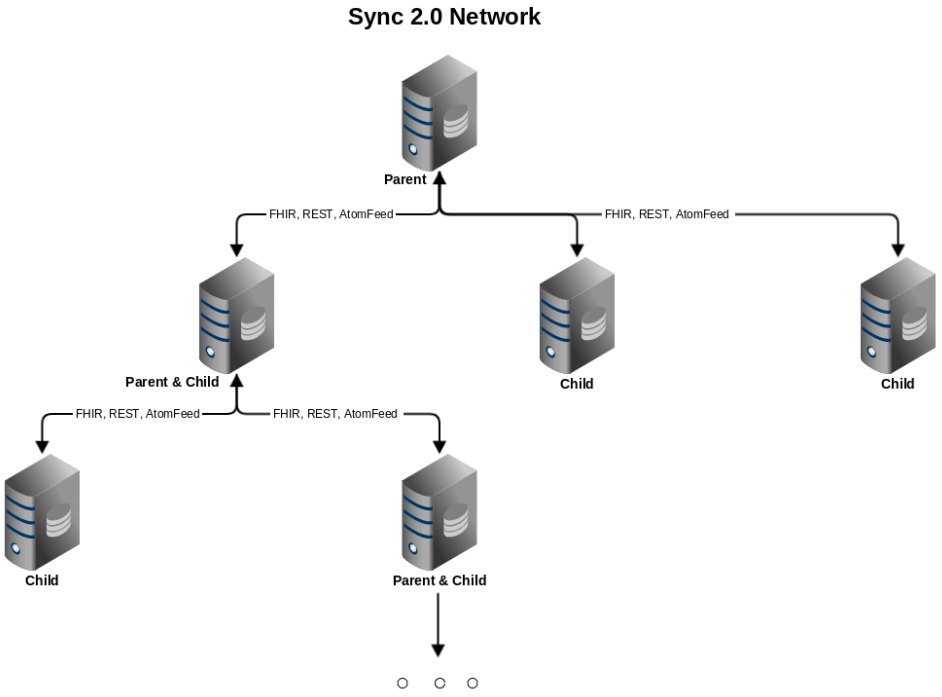
Sync 2.0 Architecture Overview

Sync 2.0 aims to use FHIR to achieve synchronization between a network of nodes. It assumes one Parent node and multiple Child nodes connected to the Parent. Each child can also act as a Parent, which allows a multi-level Sync network. In this topology, the synchronization can be done in two directions: Parent to Child and Child to Parent. Sync 2.0 will support both independently, as one-way sync is being used in the field currently by implementations adopting Sync 1.0. In both models the methods of communication will be generally the same - an atom feed will be read, then data will be retrieved and pushed, preferably using FHIR. If FHIR representations are not available for resources being transmitted, we will fall back to OpenMRS REST representations.

All synchronization will be initiated by the Children, independent from the direction that data is transmitted.



The diagram above gives an overview of a typical Sync network composed of a Parent server and multiple Child nodes synchronizing with it.

Pull from Parent

In order to keep their data in Sync, Children should read data from their Parent. This will be a three step process:

1) The Child reads through the atom feed exposed by its Parent and filters out events it is not interested in

2) For events that it is interested in, the data is pulled from Parent

3) The data is persisted by the Child

Children will be reading the feed exposed by their Parent, event by event. The Parent will expose a complete feed of events - Children will be in charge of filtering and pulling only the resources they are interested in. When they come across an event that they interpret as requiring synchronization, they will trigger retrieval of the object using the resource manager, which will deal with retrieving the object using the correct method of transmission based on the data type - either by delegating to a FHIR client, a REST client or a different client injected by implementation.

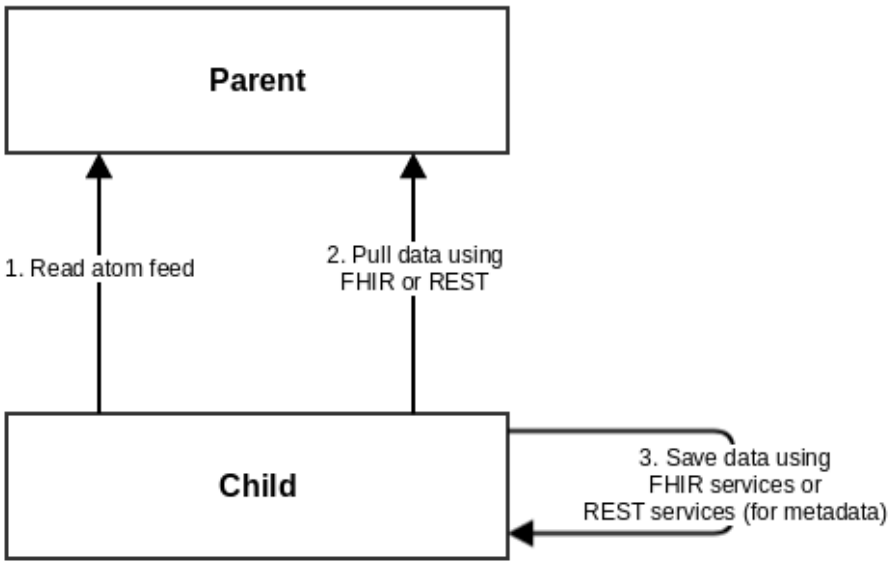
The data retrieved through FHIR, will then have to be passed to the FHIR module for saving. FHIR services from the module (example https://github.com/openmrs/openmrs-module-fhir/blob/master/api/src/main/java/org/openmrs/module/fhir/api/impl/PatientServiceImpl.java) will be capable of saving FHIR representations in OpenMRS, so Sync will only need to make use of these services and won't have to convert FHIR to OpenMRS on its own.

Metadata retrieved through the REST API will have to be inserted using conventional methods - best if it directly interfaces with OpenMRS repository classes to insert the metadata into the db.

The Child will keep track of what was synced based on its marker in the Parents feed, i.e. the marker should not be moved when a connection error happens. Retries will be controlled by configuration, it should be also possible to make a Child proceed through the feed even if a record fails to synchronize. Since the audit UI will allow manual retry of failed transactions, administrators will be able to resolve conflicts and retry transactions manually.

Pulling data from a Parent can be enabled/disabled independently from pushing data to Parent in order to allow implementations the possibility to configure one way synchronization.

The graphic below shows the steps required for the pull model:



Push from Child

The push model will be used for pushing data from Children to the Parent node. Push is also done in three steps:

1) Read own data feed, filtering out events that don't need to be synchronized with Parent

2) For events that need synchronization, read them from local endpoints either using FHIR or REST services

3) Push the representation to Parent

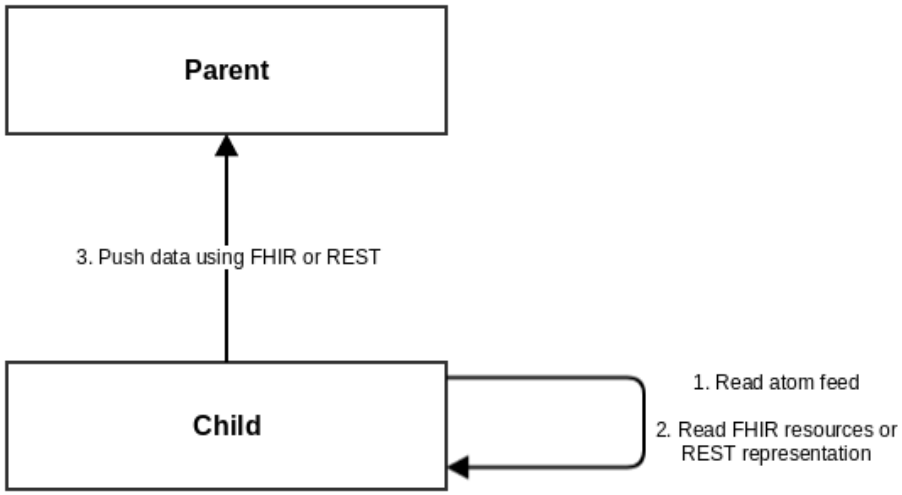
Children will publish their own atom feeds, similar to the Parent. Each Child will read it's own feed and based on what it reads from it, the Child will push the objects to Parent for synchronization.

After reading the feed, Sync will have to decide whether this is data that should be synchronized with Parent. If an object in the feed is supposed to get synchronized, the Child will first fetch it from its own FHIR server (or REST if it's metadata) through localhost and forward it to the Parent. This will allow simplicity with Sync 2.0 code - the mechanism for reading the Children feed will be similar to the Parent one. This might also allow multi leveled Sync - since Children publish their own feeds, they can also act as Parents for lower level Children.

Error handling is similar to the one described above for push.

Pushing data to Parent can be enabled/disabled independently from pulling the data from Parent in order to allow implementations the possibility to configure one way synchronization.

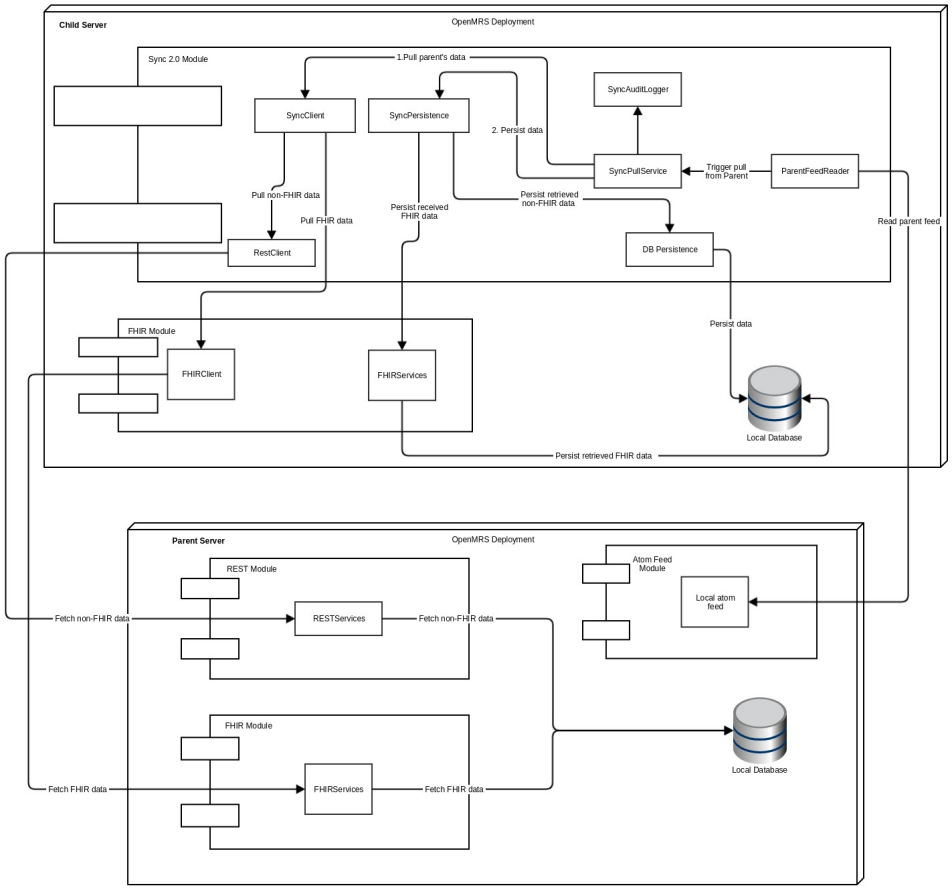
The graphic below shows the steps for the push synchronization:



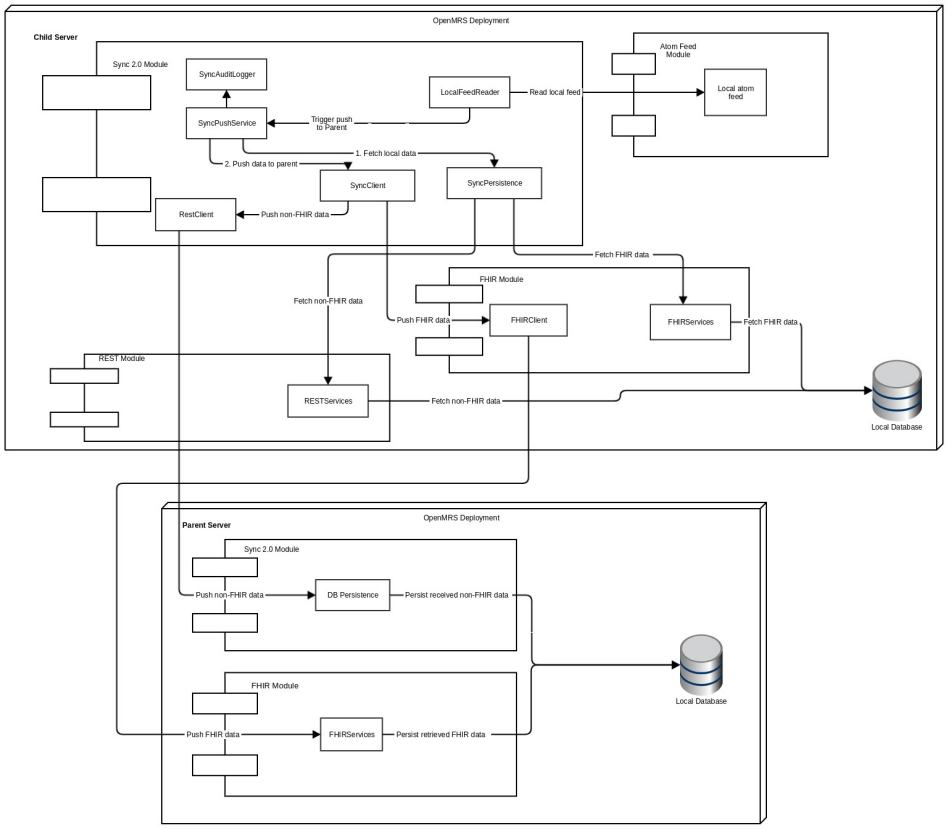
Architecture Overview

The diagram below illustrates a general overview of the Sync module architecture. Note that components might be simplified and in the real implementation split into multiple pieces. It is shown from the more interesting perspective of a Child. Note that each Child can also act as a Sync Parent.

- Architecture of pull workflow



- Architecture of push workflow



Things to note:

- Each Child can also act as a Parent
- A sync module has two feed reader instances - the local feed reader and the Parent feed reader. The local feed reader is reading the local feed and triggers a push to Parent. The Parent feed reader reads the Parent feed and triggers a pull from Parent.
- SyncPushService and SyncPullService put together the business logic for pushing and pulling
- The SyncAuditLogger is responsible for maintaining an audit of Sync operations in the database
- The SyncResourceManager delegates to either the FHIR client or a REST client, based on the data type and whether a FHIR method is possible. The underlying transfer operation should be transparent to clients.
- SyncPullService uses either FHIR services from the FHIR module or database persistence to save the data locally