TECHNICAL DOCUMENT
BAHMNI integration with DHIS2 Tracker

ThoughtWorks

Table Of Contents

1. Introduction

Patient data should flow from the Bahmni EMR system implemented at the static PSI clinic to DHIS2 cloud to generate the reports from DHIS2.

This data flow is only a one-way sync from Bahmni to DHIS2. There is no intention to push changes back from DHIS2 cloud to the clinics.

The application supports the decentralized Bahmni, where each clinic's Bahmni system interacts with a centralised DHIS server. Patient movement across the clinic is also handled in the solution. Refer here for more details.

In DHIS2 programs are configured as events with registration(Line level data), where patient registration information is captured along with the program enrollment and events.
From Bahmni we will be sending the below patient data:

1. Patient registration
2. Patient program enrollment
3. Data captured through forms in each visit

Today only HTS program data from Novo system is getting synced to DHIS2. The integration between Bahmni and DHIS2 should allow to incorporate data sync for other services which are offered in the clinic.

This technical document is a reference document which can be used to design and develop the solution for integrating Bahmni system with DHIS2 system.

Here you can checkout OpenMRS Talk about this service.

1.1 Purpose

This document provides brief overview of the Integration Service and its architecture. It is designed to assist the developers or the owner of the Bahmni Application in configuring the Integration service.

1.2 Scope

This document is with reference to the following versions of the systems:

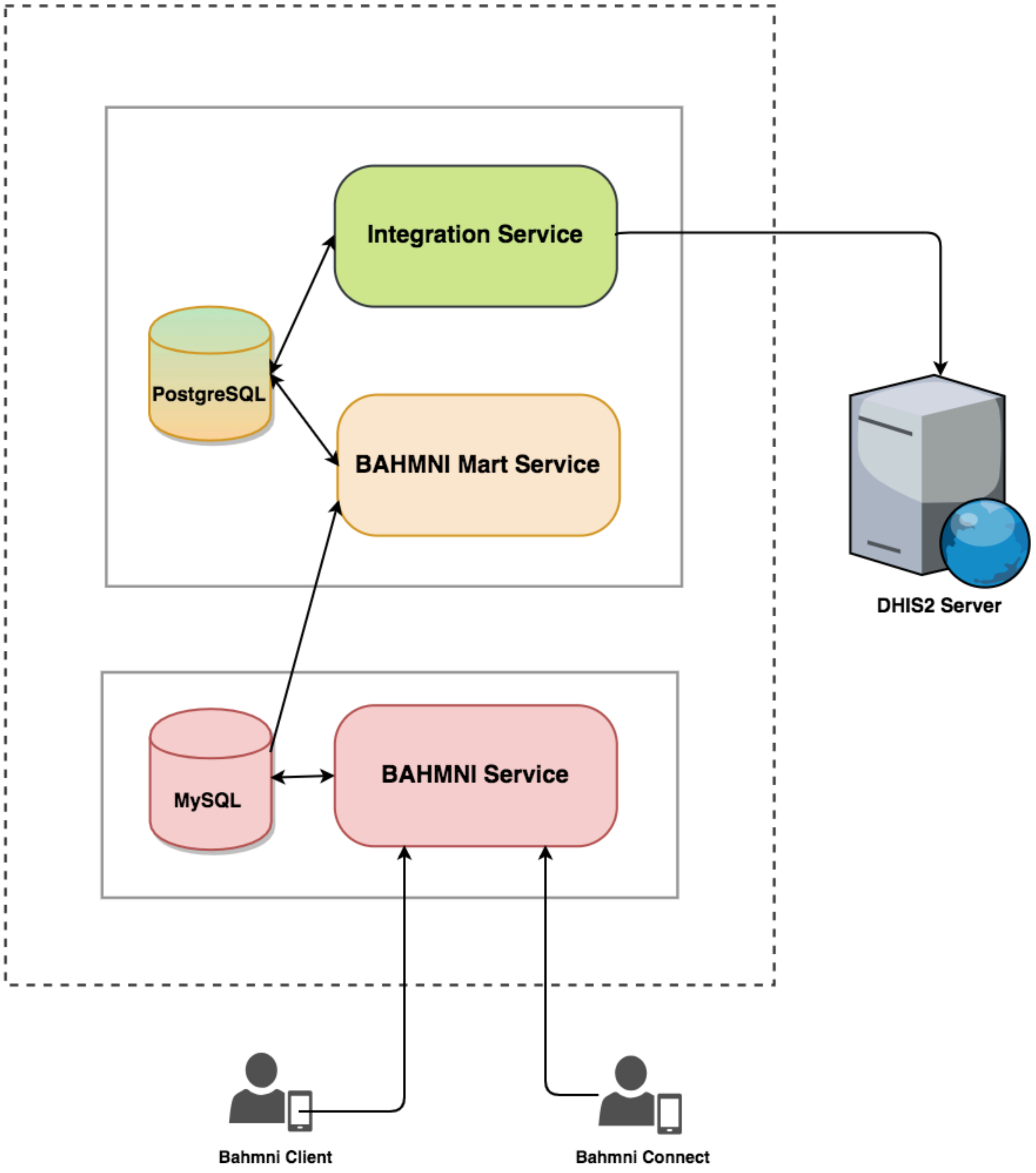
- Bahmni : **0.90**
- DHIS2: **2.30** (Build revision: **ca284eb**)
- Bahmni Mart: 1.0.0 (latest RPM which includes date created)

1.3 Definitions and Abbreviations

Abbreviated Term	Description
HIV	HIV testing service
DHIS2	District health Information System 2
PSI	Population Services International
NAH	New Africa House
UID	Unique Identification
API	Application program interface
JSON	JavaScript Object Notation
SQL	Structured Query Language
UI	User Interface
PWA	Progressive Web App
SCDF	Spring Cloud Data Flow

2. Overview

2.1 Architecture



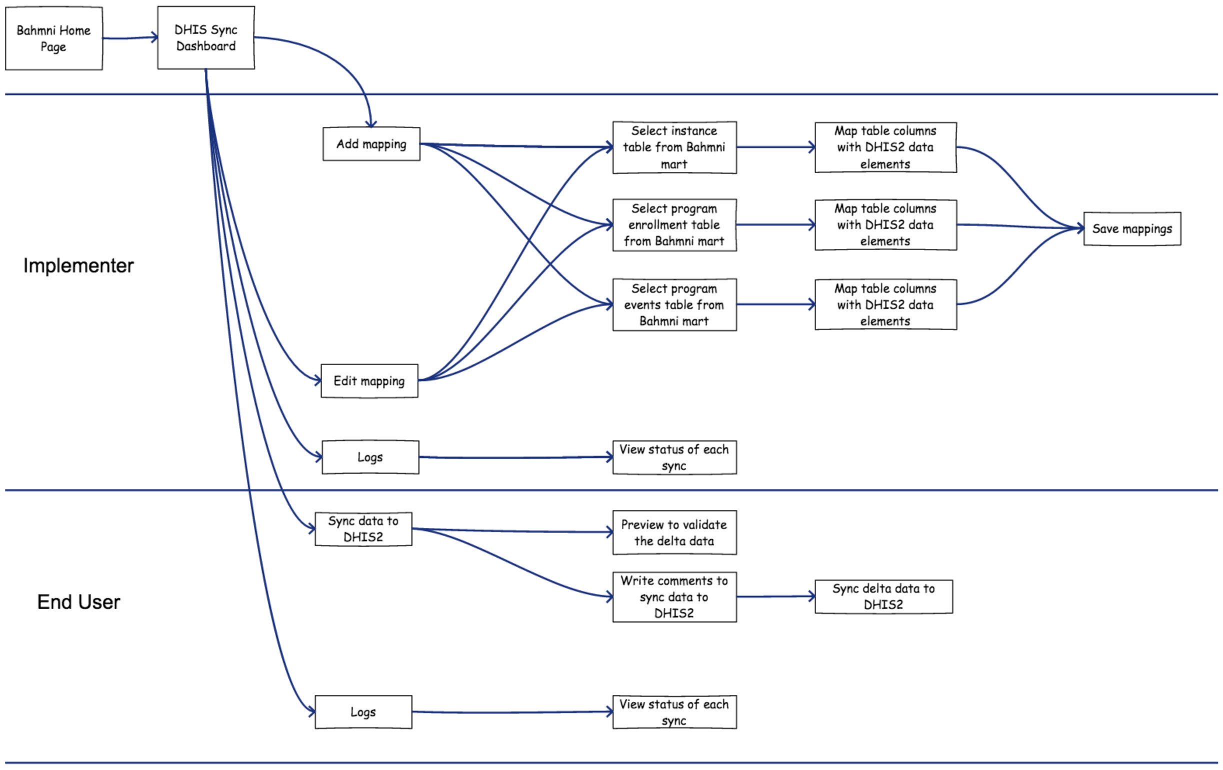
This architecture involves the following entities:

- Bahmni Client: Laptops/Tablets used by the providers to access Bahmni.
- Bahmni Connect (PWA) : An app which allows a user to access Bahmni in areas with limited internet connectivity. The provider can later come into an area with connectivity to a Bahmni server, and synchronize the PWA on their Android device/Chromebook with the Bahmni server.
- Bahmni Server: The open source EMR server and a full Hospital Management System. It maintains its own (MySQL) database which is the source of truth.
- Bahmni Mart Service: A standalone application which will create various tables in an analytics DB (PostgreSQL) from Bahmni/Openmrs hierarchical DB (MySQL).
- Integration Service: An application (plugged into SCDF, dependent on Bahmni Mart) to perform data sync from Bahmni server to DHIS2 server. It would maintain tables in the same analytics DB (PostgreSQL) as that of Bahmni Mart. These tables would persist information for translating respective Bahmni data to DHIS2 data.
- DHIS2 Server: A web-based open-source information system with awesome visualization. It maintains its own (PostgreSQL) database which is used for the reporting and analytics.

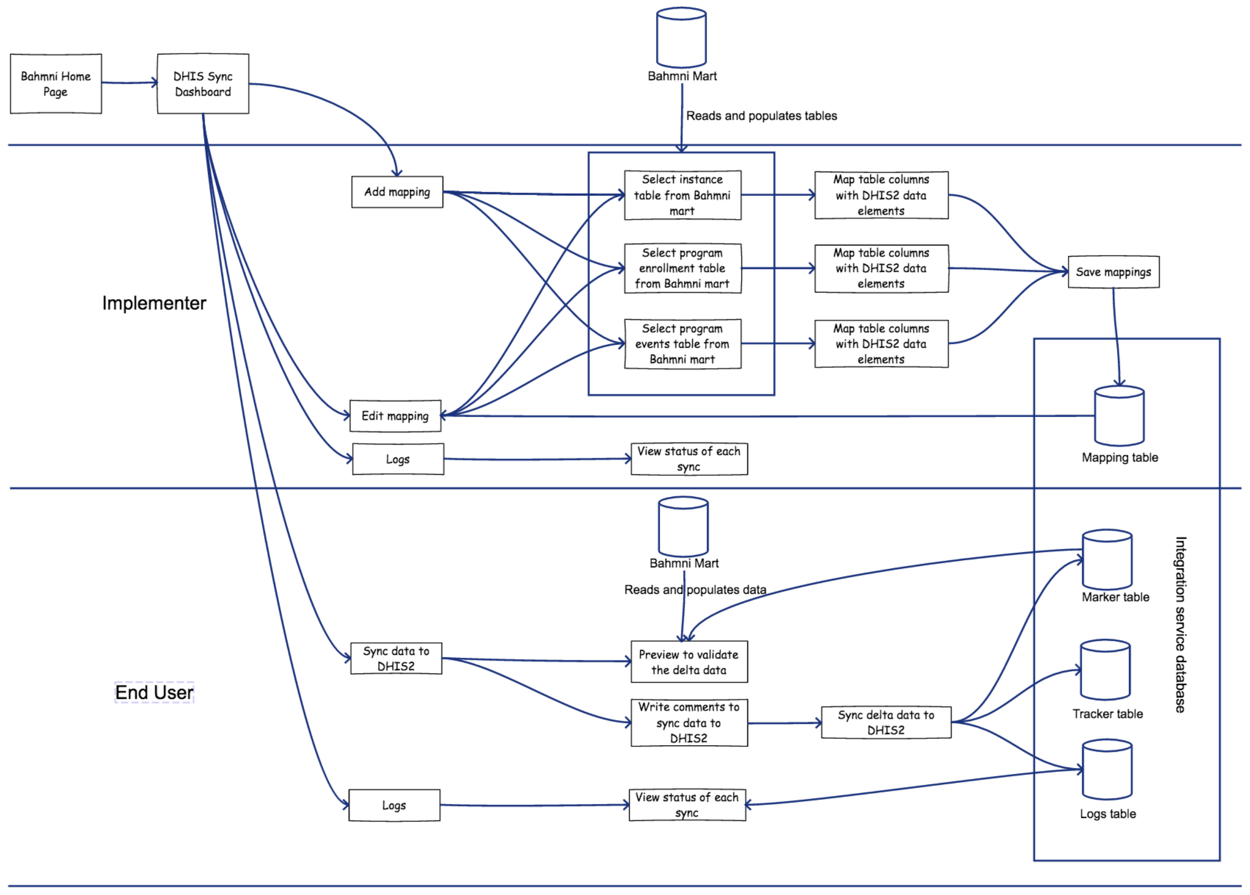
2.2 Business Flow

Below are the business process flow diagram at Level 0 and Level 1 to show the interaction of user at various points.

1. Business process flow - Level 0


Bahmni Home Page
DHIS Sync Dashboard
Implementer
Add mapping
Edit mapping
Logs
Select instance table from Bahmni mart
Select program enrollment table from Bahmni mart
Select program events table from Bahmni mart
Map table columns with DHIS2 data elements
Map table columns with DHIS2 data elements
Map table columns with DHIS2 data elements
Save mappings
View status of each sync
End User
Sync data to DHIS2
Preview to validate the delta data
Write comments to sync data to DHIS2
Sync delta data to DHIS2
Logs
View status of each sync

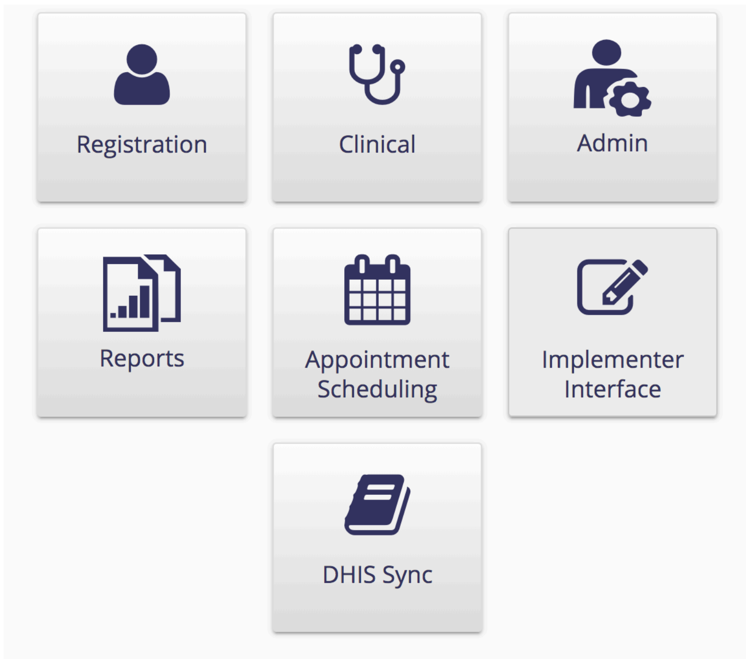
2. Business process flow - Level 1


Bahmni Home Page
DHIS Sync Dashboard
Bahmni Mart
Reads and populates tables
Implementer
Add mapping
Select instance table from Bahmni mart
Select program enrollment table from Bahmni mart
Select program events table from Bahmni mart
Map table columns with DHIS2 data elements
Map table columns with DHIS2 data elements
Map table columns with DHIS2 data elements
Save mappings
Edit mapping
Logs
View status of each sync
Mapping table
Integration service database
End User
Bahmni Mart
Reads and populates data
Marker table
Sync data to DHIS2
Preview to validate the delta data
Write comments to sync data to DHIS2
Sync delta data to DHIS2
Tracker table
Logs
View status of each sync
Logs table

3. Bahmni-DHIS2 Integration

3.1 User Interface

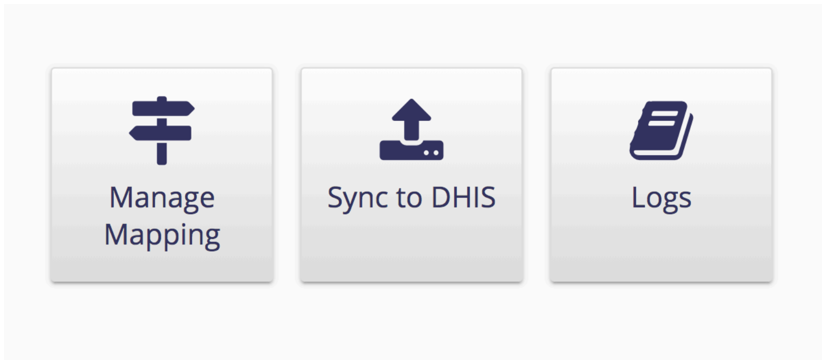
The UI module of the Integration Service is called ‘DHIS2 sync’ module. It will be available on the Bahmni home page as shown below. Only privileged users would have access to this module.



Bahmni Home Page

On clicking the ‘DHIS2 sync’’ module the user would be redirected to the ‘DHIS2 sync’ landing page. From here the following functionalities would be available:

1. Manage Mapping (Add/Edit)
2. Sync to DHIS
3. Logs



‘DHIS2 sync’ landing page

3.1.1 Manage mapping

There is a need to map the Bahmni table to DHIS2 data elements. This information will be stored as mappings (JSON format) which will be used to sync the data to DHIS2.

User needs to provide mapping information for some categories. User will be asked to select the database table from a dropdown for each of the above three categories. Based on the selection, it will populate all the table columns in the form. The user needs to enter the DHIS2 data element UIDs in the editable text boxes against the columns.

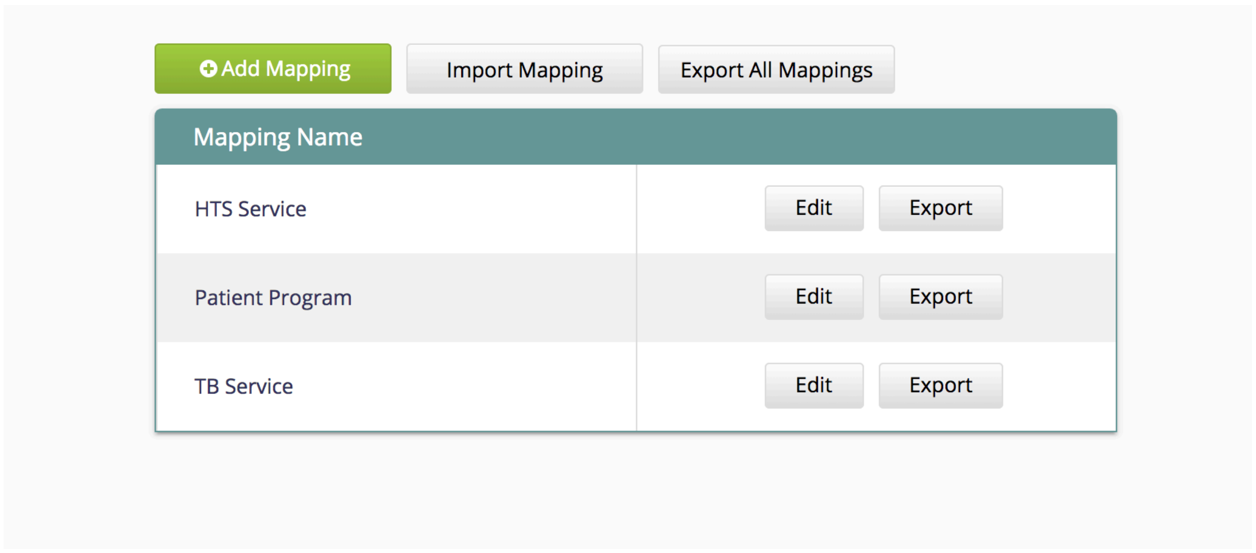
The categories for which the mapping information is required are:

1. Patient registration
    - Herein, the user has to enter at least one column (person attribute) mapping.
    - The user also has the benefit of (optionally) marking one or more person attributes as ‘searchable’ and ‘configurable’.
    - Before syncing a new patient to DHIS it would query the DHIS server for the presence of this new patient using the ‘searchable’ fields to avoid the duplication of the patient in DHIS.
    - If the search doesn’t return any UIDs then it will sync the patient as a NEW patient. If the search returns a single UID then it would sync the patient data as UPDATE patient using the UID.
    - If the search returns multiple results, then the ‘comparable’ fields will be used to filter down the result set.
    - If the filtered result returns no UIDs then it will sync the patient as a NEW patient. If the search returns a single UID then it would sync the patient data as UPDATE patient using the UID.
    - Although if the search returns multiple UIDs still, then the result set is ignored and the patient is synced as a NEW patient.

2. Program enrollment
    - Herein, the user has to also answer a question: Do you want to open latest completed enrollment?
    - For a given patient, some programs like HIV would have only single enrollment in the lifetime. Whereas, the same patient can have multiple enrollments for a single program like Pregnancy. To handle both types of programs a configuration is available in the mapping. This configuration will decide to Open an existing closed enrolment or create a new enrolment in DHIS.
    - Detailed impact of setting this configuration is shown in the appendix.

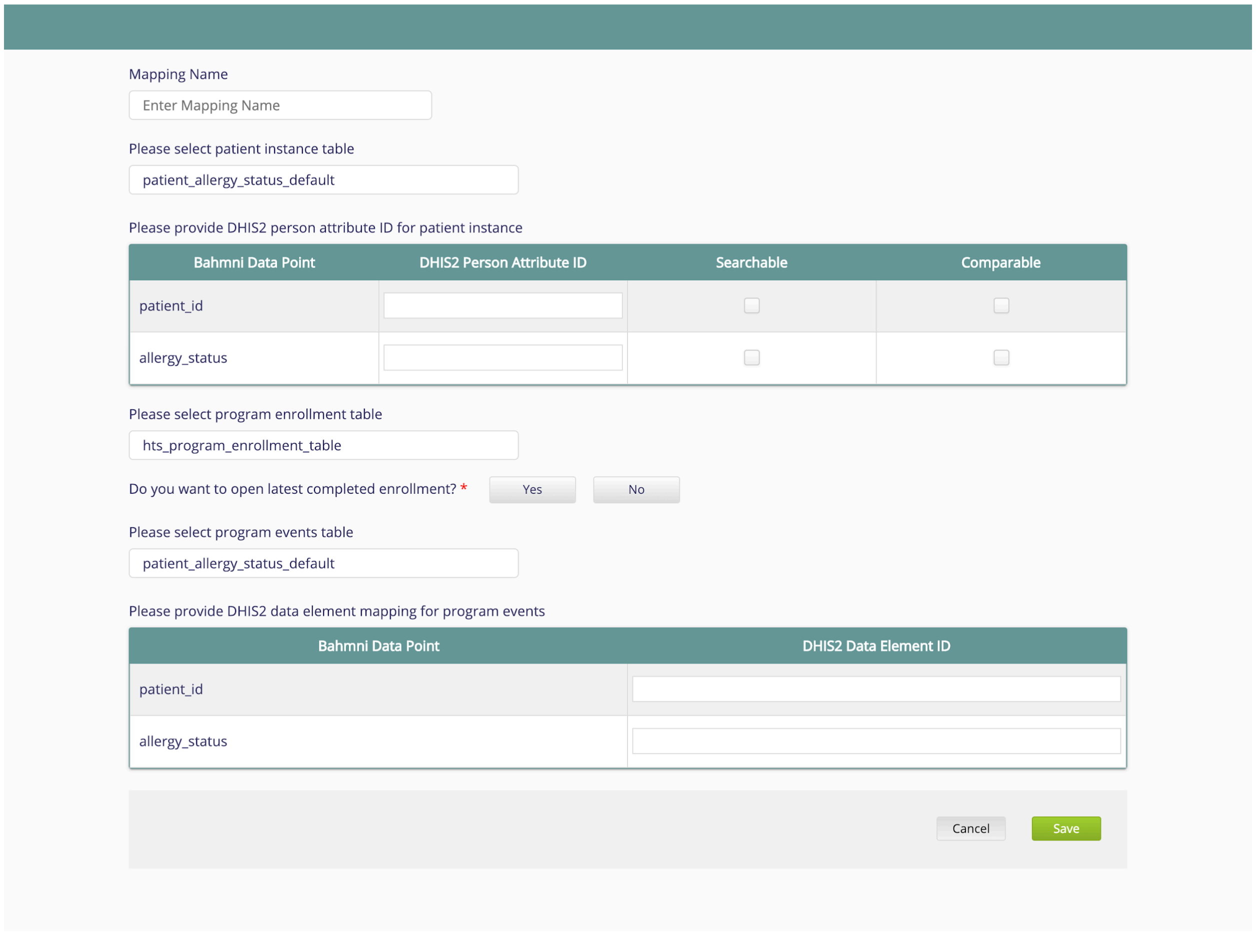
3. Program event
    - Herein, at least one data element has to be entered.

Below are the screenshots:



Mapping Name	
HTS Service	Edit Export
Patient Program	Edit Export
TB Service	Edit Export

DHIS2 Mapping Dashboard


Mapping Name

Enter Mapping Name

Please select patient instance table

patient_allergy_status_default

Please provide DHIS2 person attribute ID for patient instance

Bahmni Data Point	DHIS2 Person Attribute ID	Searchable	Comparable
patient_id		☐	☐
allergy_status		☐	☐

Please select program enrollment table

hts_program_enrollment_table

Do you want to open latest completed enrollment? * Yes No

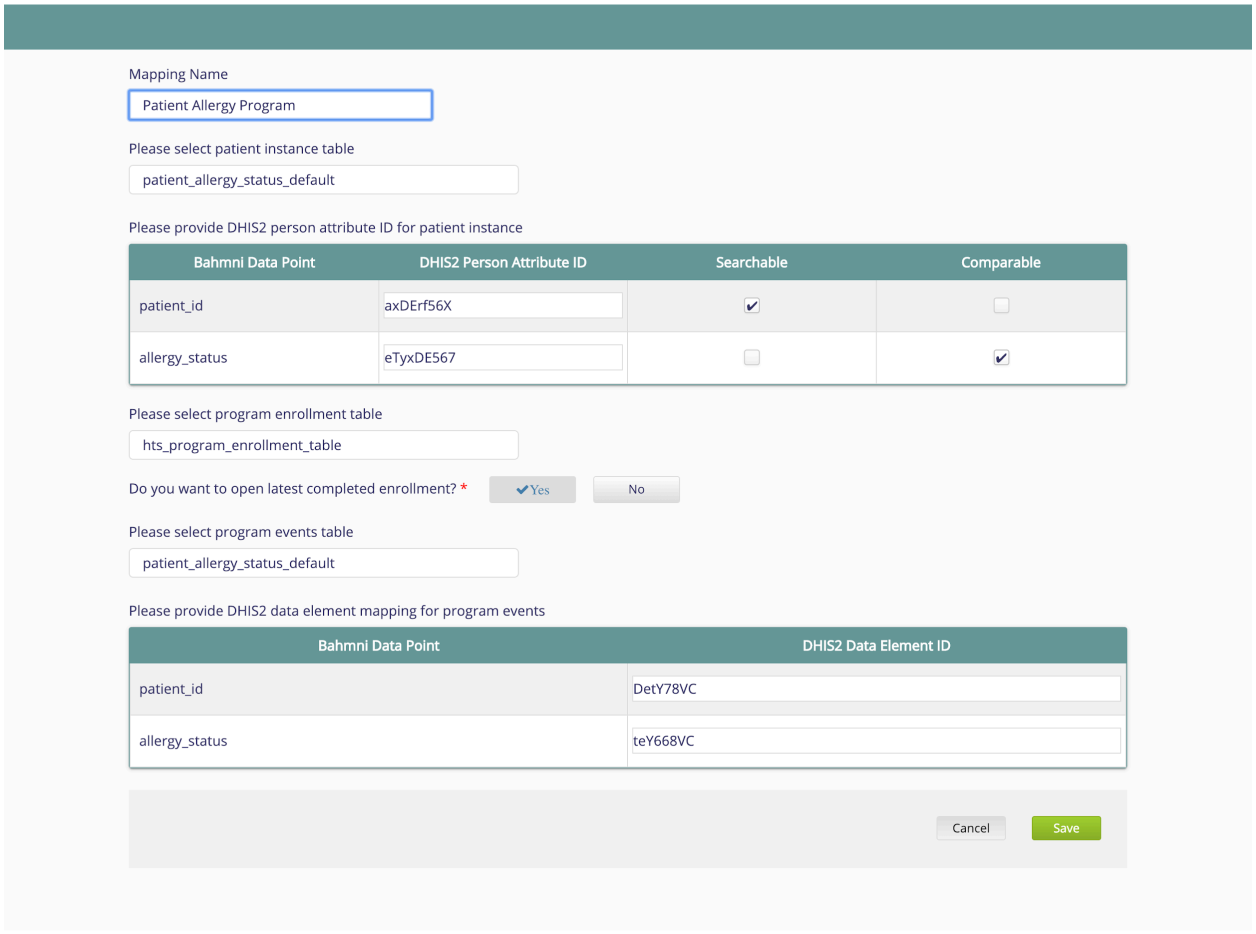
Please select program events table

patient_allergy_status_default

Please provide DHIS2 data element mapping for program events

Bahmni Data Point	DHIS2 Data Element ID
patient_id	
allergy_status	

Adding new mapping



Mapping Name

Patient Allergy Program

Please select patient instance table

patient_allergy_status_default

Please provide DHIS2 person attribute ID for patient instance

Bahmni Data Point	DHIS2 Person Attribute ID	Searchable	Comparable
patient_id	axDErf56X	✓	
allergy_status	eTyxDE567		✓

Please select program enrollment table

hts_program_enrollment_table

Do you want to open latest completed enrollment? * [✓ Yes] No

Please select program events table

patient_allergy_status_default

Please provide DHIS2 data element mapping for program events

Bahmni Data Point	DHIS2 Data Element ID
patient_id	DetY78VC
allergy_status	teY668VC

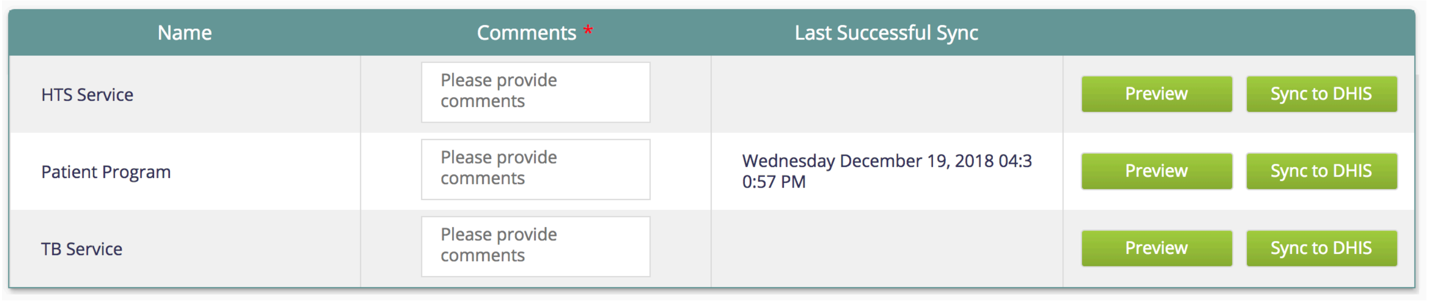
Editing existing mapping

3.1.2 Sync Data to DHIS2

From here the following actions can be performed:

1. Preview data
2. View last successful sync date
3. Send data to DHIS2

User will be validating the data before syncing to DHIS2. For this purpose user could click on ‘Preview’ button to view and validate the delta data which is yet to synced to DHIS2. Once the data is validated user would click on ‘Send to DHIS2’ to sync the delta data to DHIS2.

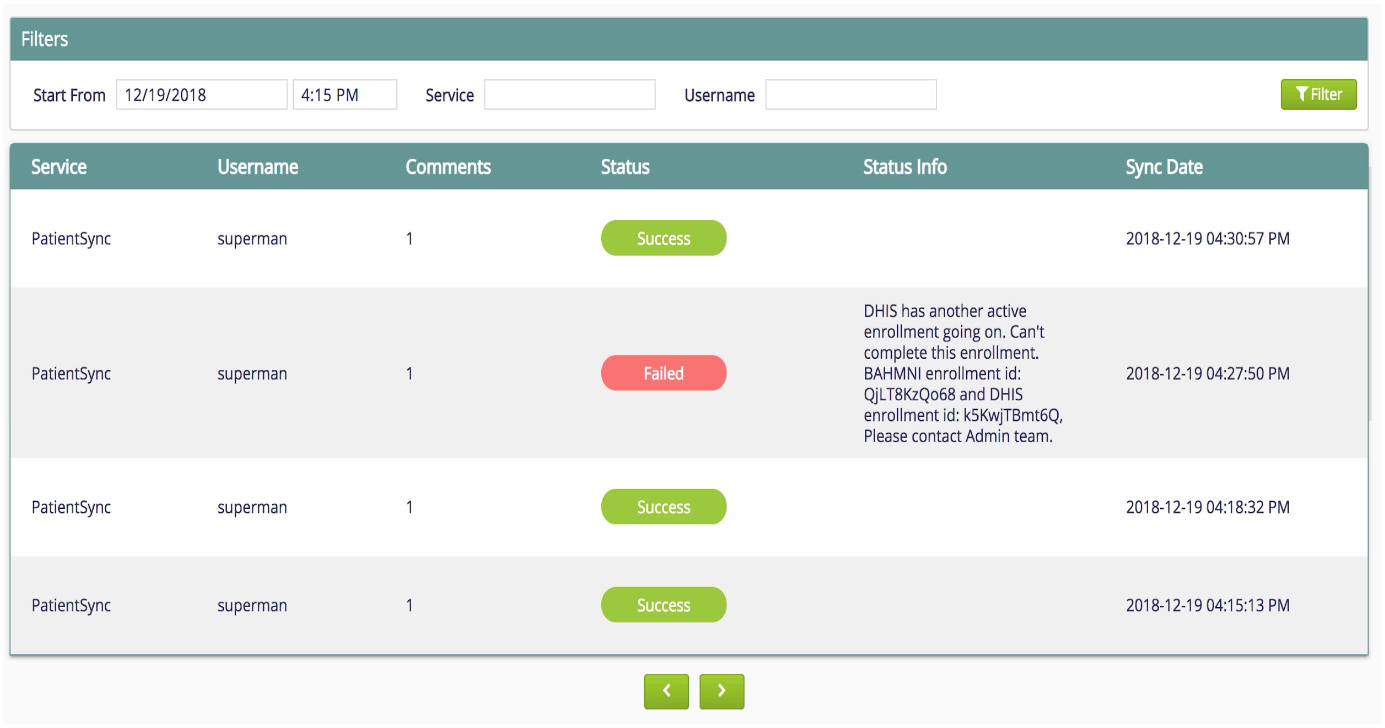


Name	Comments *	Last Successful Sync	
HTS Service	Please provide comments		Preview / Sync to DHIS
Patient Program	Please provide comments	Wednesday December 19, 2018 04:30:57 PM	Preview / Sync to DHIS
TB Service	Please provide comments		Preview / Sync to DHIS

Sync data to DHIS

3.1.3 Logs

For every data sync to DHIS2 a log entry will be made along with the sync status and additional information.



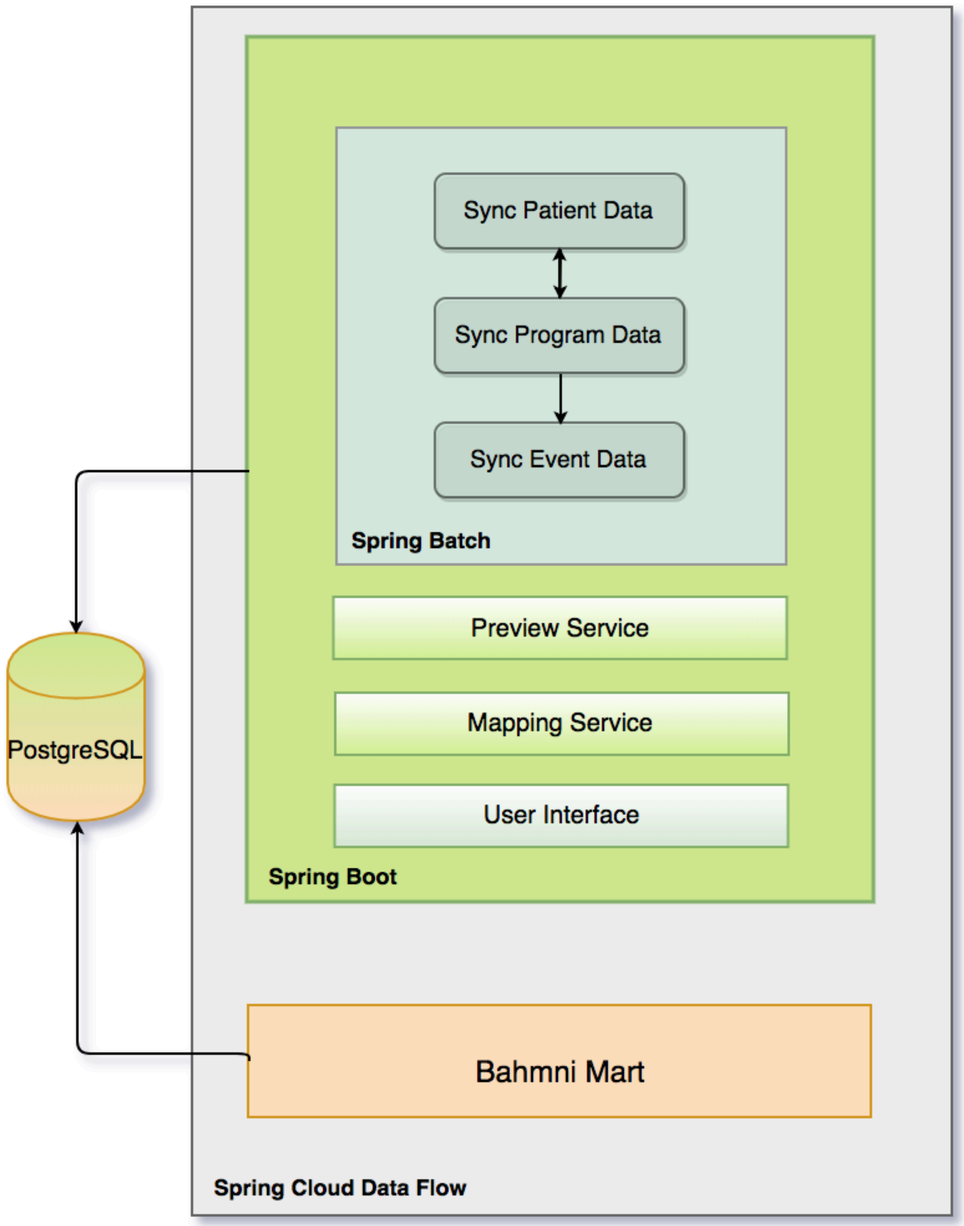
Filters

Start From 12/19/2018 4:15 PM Service Username Filter

Service	Username	Comments	Status	Status Info	Sync Date
PatientSync	superman	1	Success		2018-12-19 04:30:57 PM
PatientSync	superman	1	Failed	DHIS has another active enrollment going on. Can't complete this enrollment. BAHMNI enrollment id: QjLT8KzQo68 and DHIS enrollment id: k5KwjTBmt6Q, Please contact Admin team.	2018-12-19 04:27:50 PM
PatientSync	superman	1	Success		2018-12-19 04:18:32 PM
PatientSync	superman	1	Success		2018-12-19 04:15:13 PM

Logs for DHIS2 sync activity

3.2 Application Components



A Spring Cloud Data Flow (SCDF) server is configured with Bahmni Mart application. This is an existing platform. On this platform, a new Spring Boot Application ‘Bahmni-DHIS2 Integration’ application will be created to sync data from Bahmni to DHIS2. This application would share the Bahmni Mart database and would be configured by SCDF dashboard.

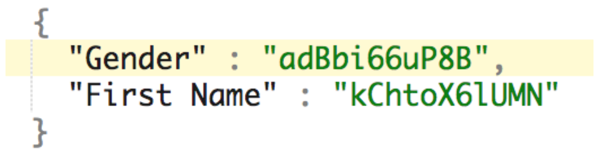
This application would contain several services and an embedded Spring Batch Application. The services will be used to respond to the REST API requests. The Spring Batch application would sync the data in 3 steps as there are 3 types of data:

1. Patient Data: All the attributes of the patient.
2. Program Data: All the details of the program in which the patient is registered.
3. Event Data: All the detailed observations per patient captured as part of the Program.

The detailed API calls for each of these data syncs is provided in next section here.

3.2.1 Mapping Service

The smallest unit in DHIS2 is a data element (analogous to Bahmni's concept). And each data element is identified by a unique UID. There is a need to map the Bahmni Mart tables' column to DHIS2 data elements. The 'Mapping Service' would maintain this mapping information for each program. It would be captured in JSON format as shown below and saved in 'Mapping Table'.



```
{
  "Gender" : "adBbi66uP8B",
  "First Name" : "kChtoX6lUMN"
}
```

Mapping JSON example

As in the given example, the key name *Gender* in Bahmni would be represented as *adBbi66uP8B* in DHIS2. This conversion should be configurable as it varies between implementations. UI is provided in order to define these mappings. For detailed Mapping table format refer here

3.2.2 Preview Service

Preview service would enable user to view the (delta) data which is yet to be synced. This would be helpful for the user to validating the data before syncing. Only the delta data that has not been synced previously to DHIS2 will be shown.
This service uses Atom feed service and has its own set of Marker table(s).
When user clicks on the 'Preview' button then the Preview service will get a REST API request and respond with all the data being synced:

1. It reads the last_synced_date from the Marker table
2. Based on this date the Preview Service will request Bahmni Mart to get all the required information (uses SQL to gather this information)

3.2.3 Sync Data Batch

In the 'DHIS2 Sync' module's landing page, user would click on 'Sync data to DHIS2' which will be redirected to another page. Here once the user clicks on 'Send data to DHIS2' then this Spring Batch application gets invoked. It syncs data from Bahmni to DHIS2 in a bulk job.

The application would execute the following steps to sync data:

1. Sync Patient Data
    A. Fetch the delta patient data (in chunks) using the Marker and Mart Tables
    B. Join the patient (Bahmni) ID with Patient (DHIS2) UID from the Instance_Tracker table. The patients whose UID is *not* available in the table are the NEW patients. The patients who have a UID recorded in the table are UPDATED patients.
    C. Get the patient list from the DHIS based on the searchable & comparable attributes. Based on the final resultset update the patient UIDs in this delta data.
    D. Get the patient mapping from Mapping table.
    E. Augment this patient data with the mapping information to formulate the REST API request body to be sent to DHIS2.
    F. Send patient data (NEW & UPDATED together) to DHIS2.
    G. Update Instance_tracker table based on the response.

2. Get Enrolments' information from DHIS
    A. Fetch the delta enrolment data and its events (in chunks) using the Marker & Mart Tables.
    B. Fetch all the enrollments' details for the patients whose enrollments are yet to be synced.
    C. Depending on the configuration set in the mapping to open a closed enrollment, update the UIDs in-memory for the enrollments.

3. Sync Newly Completed Enrollments with its Events
    A. Fetch all the enrollments' details for the patients whose enrollments are yet to be synced with status as COMPLETED.
    B. Join the enrolment data with Program-Name, Program-Date from the Enrolment_Tracker table (based on the Patient UID from the previous step). The enrolments whose Enrolment UID is *not* available in the table are the NEW enrolments.
    C. Join the event data with event (DHIS2) UID, Visit-ID from the Event_Tracker table.The events whose UID is *not* available in the table are the NEW events.
    D. Get the Program enrollment mapping and Event mapping from Mapping table.
    E. Augment this patient enrollment data with the mapping information to formulate the REST API request body to be sent to DHIS2.
    F. Send NEW enrollment data (with status as ACTIVE) with its events to DHIS2.
    G. Update Enrollments_tracker & Event_tracker tables based on the response.
    H. Send NEW enrollment data (with status as COMPLETED) to DHIS2.
    I. Update Enrollments_tracker based on the response.

4. Sync Updated Completed Enrollments with its Events

A. Fetch all the enrollments' details for the patients whose enrollments are yet to be synced with status as COMPLETED.
B. Join the enrolment data with Enrolment (DHIS2) UID, Program-Name, Program-Date from the Enrolment_Tracker table (based on the Patient UID from the previous step). The enrolments who have a UID recorded in the table are UPDATED enrolments.
C. Join the event data with event (DHIS2) UID, Visit-ID from the Event_Tracker table. The events whose UID is not available in the table are the NEW events. The events who have a UID recorded in the table are UPDATED events.
D. Get the Program enrollment mapping and Event mapping from Mapping table.
E. Augment this patient enrollment data with the mapping information to formulate the REST API request body to be sent to DHIS2.
F. Send UPDATED enrollment data with its events to DHIS2.
G. Update Enrollments_tracker & Event_tracker tables based on the response.

5. Sync Newly Active Enrollments with its Events
    A. Fetch all the enrollments' details for the patients whose enrollments are yet to be synced with status as ACTIVE.
    B. Join the enrolment data with Program-Name, Program-Date from the Enrolment_Tracker table (based on the Patient UID from the previous step). The enrolments whose Enrolment UID is *not* available in the table are the NEW enrolments.
    C. Join the event data with event (DHIS2) UID, Visit-ID from the Event_Tracker table.The events whose UID is *not* available in the table are the NEW events.
    D. Get the Program enrollment mapping and Event mapping from Mapping table.
    E. Augment this patient enrollment data with the mapping information to formulate the REST API request body to be sent to DHIS2.
    F. Send NEW enrollment data with its events to DHIS2.
    G. Update Enrollments_tracker & Event_tracker tables based on the response.

6. Sync Updated Active Enrollments with its Events
    A. Fetch all the enrollments' details for the patients whose enrollments are yet to be synced with status as ACTIVE.
    B. Join the enrolment data with Enrolment (DHIS2) UID, Program-Name, Program-Date from the Enrolment_Tracker table (based on the Patient UID from the previous step). The enrolments who have a UID recorded in the table are UPDATED enrolments.
    C. Join the event data with event (DHIS2) UID, Visit-ID from the Event_Tracker table. The events whose UID is not available in the table are the NEW events. The events who have a UID recorded in the table are UPDATED events.
    D. Get the Program enrollment mapping and Event mapping from Mapping table.
    E. Augment this patient enrollment data with the mapping information to formulate the REST API request body to be sent to DHIS2.

F. Send UPDATED enrollment data with its events to DHIS2.
G. Update Enrollments_tracker & Event_tracker tables based on the response.
H. Update Marker table with the sync status.

In the case of failure of either of the above steps:

1) The Marker Table would not be updated.
2) The sync would be aborted without moving on with the rest of the steps.
3) The Log table would be updated with the sync status.

3.3 DHIS2 sync API

To sync the data for any program the following 3 API calls have to be made to DHIS2:

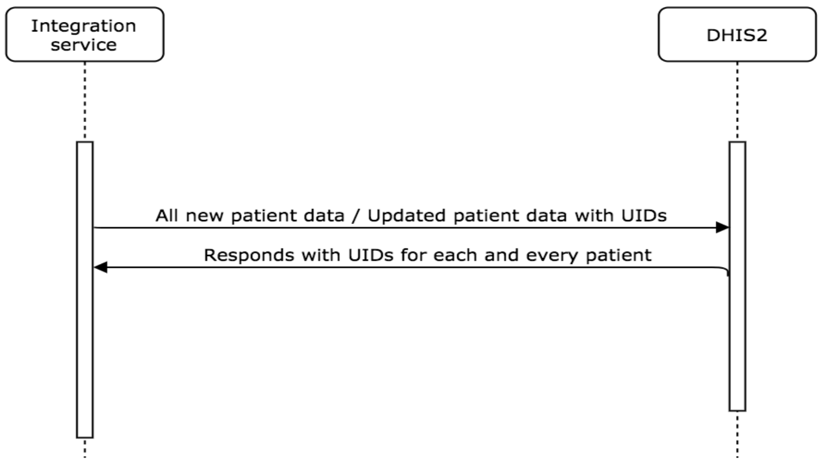
1) Creating a new tracked entity instance
2) Enrolling a tracked entity instance into a program
3) Creating an event for a tracked entity instance for a given program

3.3.1 Creating a new tracked entity instance

A tracked entity Instance in DHIS2 world is analogous to a new patient in Bahmni world. For creating a new patient in the system, you will be working with the *trackedEntityInstances* resource. The 'TrackedEntityType' represents tracking a 'Person' entity.

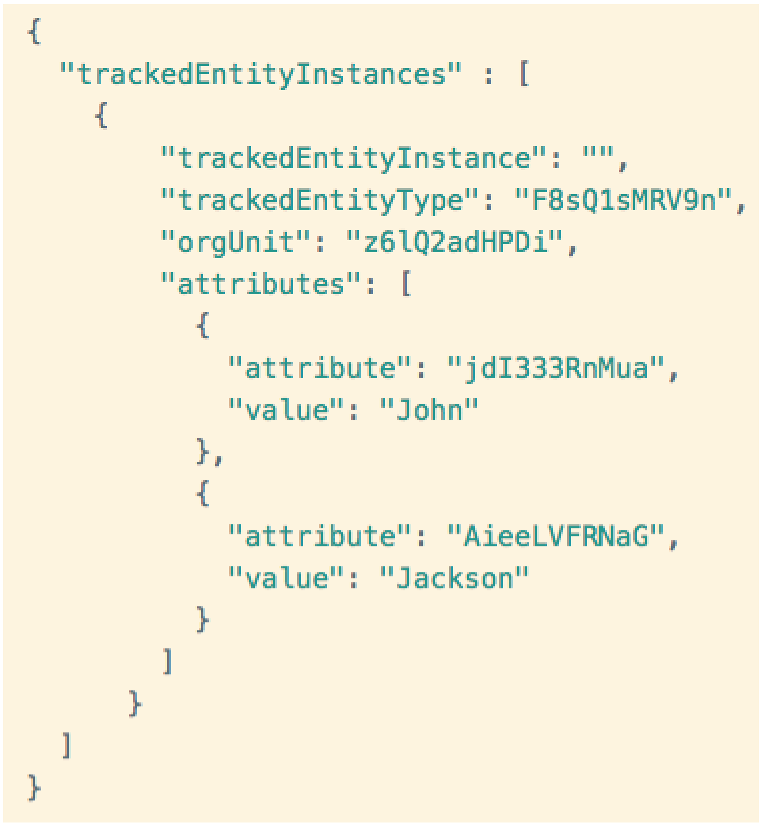
<u>NEW patient:</u> Every successful creation of tracked entity Instance would be associated with a unique UID in DHIS2. The same is available under 'reference' in the response body.

<u>UPDATE patient:</u> For updating any patient details the same UID must be sent as "trackedEntityInstance" to update its details in DHIS2.


Integration service
DHIS2
All new patient data / Updated patient data with UIDs
Responds with UIDs for each and every patient

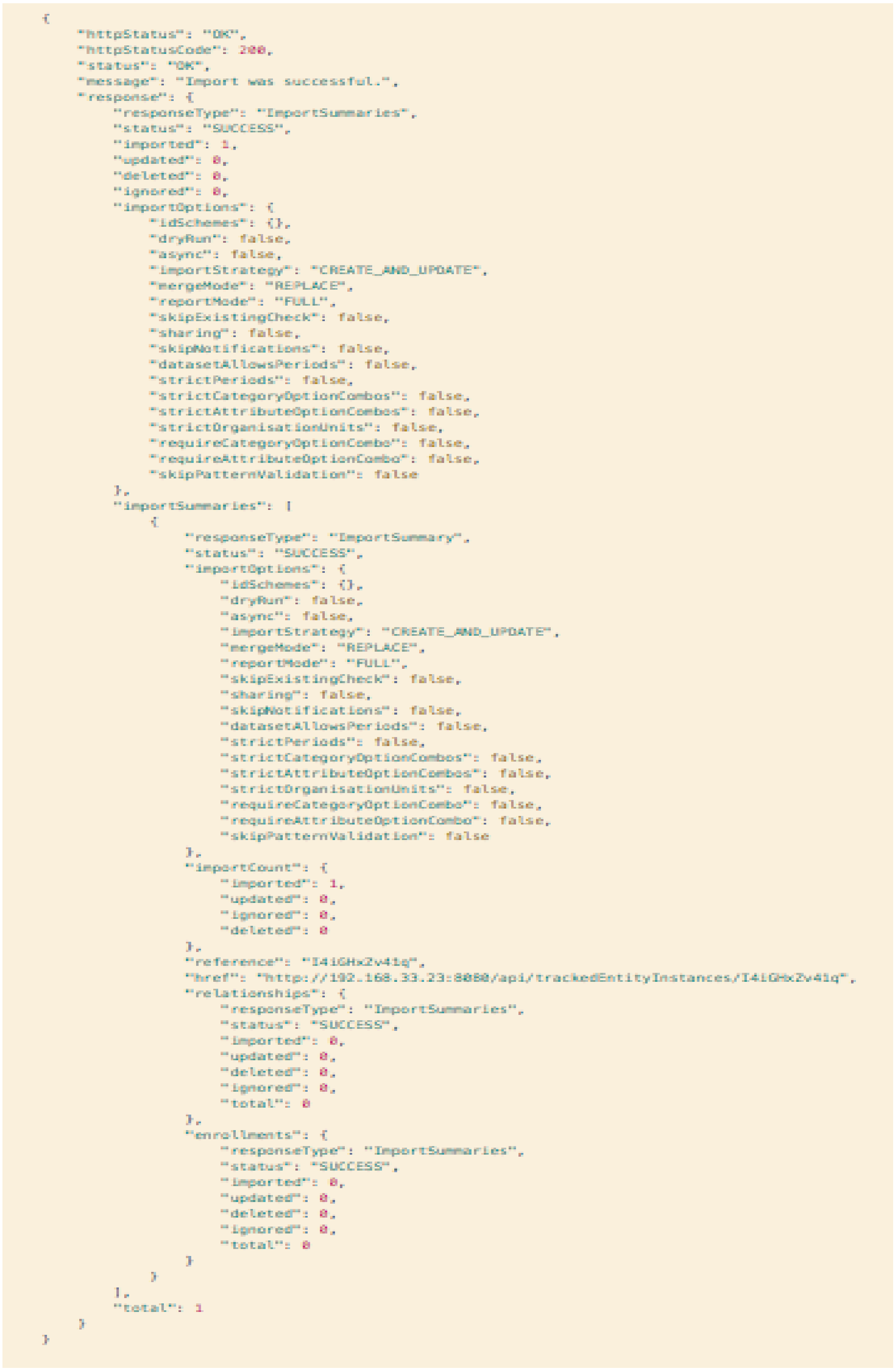
URL: http://<DHIS2-SERVER>/api/trackedEntityInstances?strategy=CREATE_AND_UPDATE

Request Body Sample:



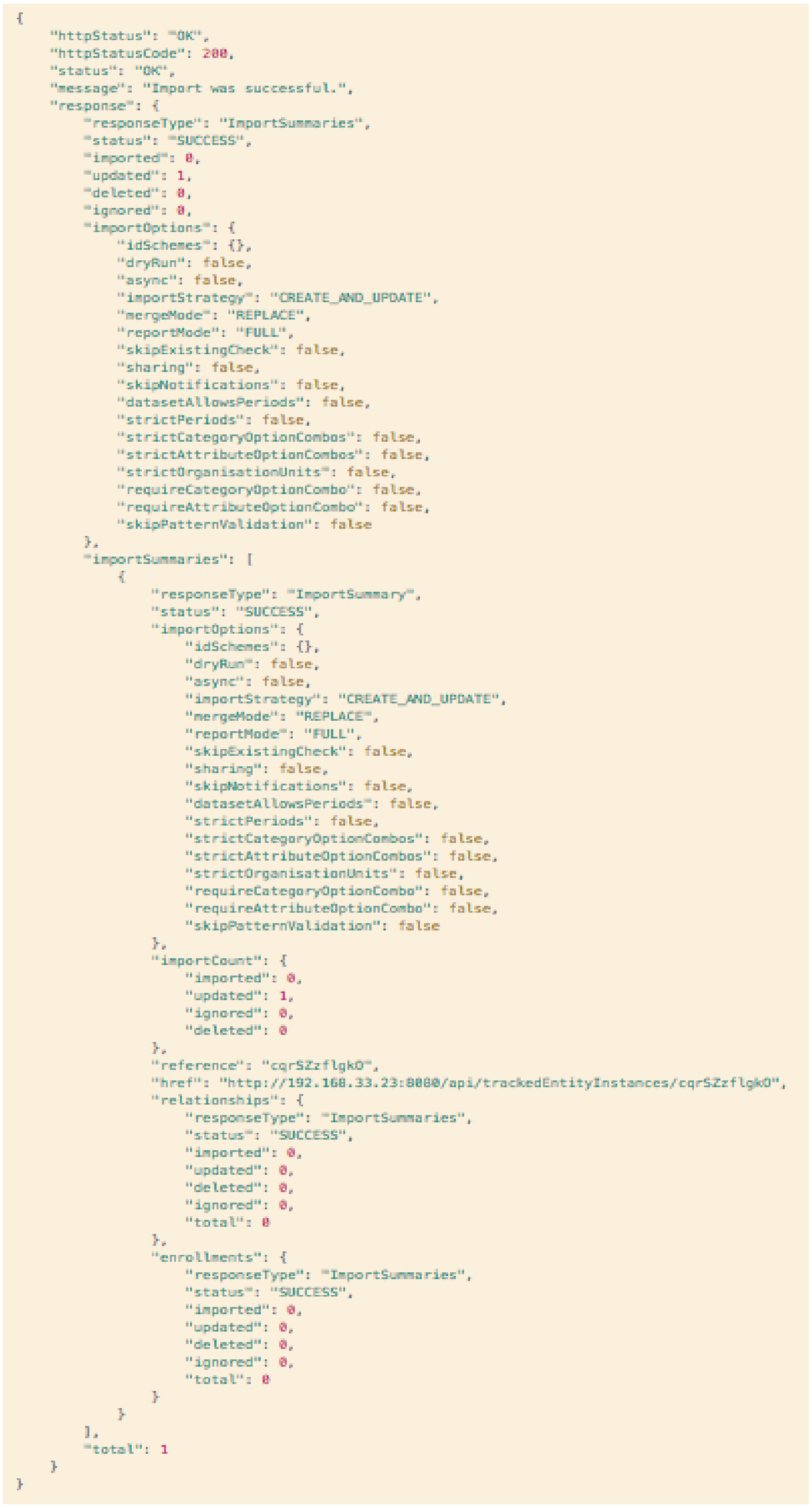
```
{
  "trackedEntityInstances" : [
    {
        "trackedEntityInstance": "",
        "trackedEntityType": "F8sQ1sMRV9n",
        "orgUnit": "z6lQ2adHPDi",
        "attributes": [
          {
            "attribute": "jdI333RnMua",
            "value": "John"
          },
          {
            "attribute": "AieeLVFRNaG",
            "value": "Jackson"
          }
        ]
      }
  ]
}
```

Response Body (Success) Sample:



```
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 1,
        "updated": 0,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {
            "idSchemes": {},
            "dryRun": false,
            "async": false,
            "importStrategy": "CREATE_AND_UPDATE",
            "mergeMode": "REPLACE",
            "reportMode": "FULL",
            "skipExistingCheck": false,
            "sharing": false,
            "skipNotifications": false,
            "datasetAllowsPeriods": false,
            "strictPeriods": false,
            "strictCategoryOptionCombos": false,
            "strictAttributeOptionCombos": false,
            "strictOrganisationUnits": false,
            "requireCategoryOptionCombo": false,
            "requireAttributeOptionCombo": false,
            "skipPatternValidation": false
        },
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {
                    "idSchemes": {},
                    "dryRun": false,
                    "async": false,
                    "importStrategy": "CREATE_AND_UPDATE",
                    "mergeMode": "REPLACE",
                    "reportMode": "FULL",
                    "skipExistingCheck": false,
                    "sharing": false,
                    "skipNotifications": false,
                    "datasetAllowsPeriods": false,
                    "strictPeriods": false,
                    "strictCategoryOptionCombos": false,
                    "strictAttributeOptionCombos": false,
                    "strictOrganisationUnits": false,
                    "requireCategoryOptionCombo": false,
                    "requireAttributeOptionCombo": false,
                    "skipPatternValidation": false
                },
                "importCount": {
                    "imported": 1,
                    "updated": 0,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "I4iGHx2v41q",
                "href": "http://192.168.33.23:8080/api/trackedEntityInstances/I4iGHx2v41q",
                "relationships": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "total": 0
                },
                "enrollments": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "total": 0
                }
            }
        ],
        "total": 1
    }
}
```

Response Body (Updated) Sample:



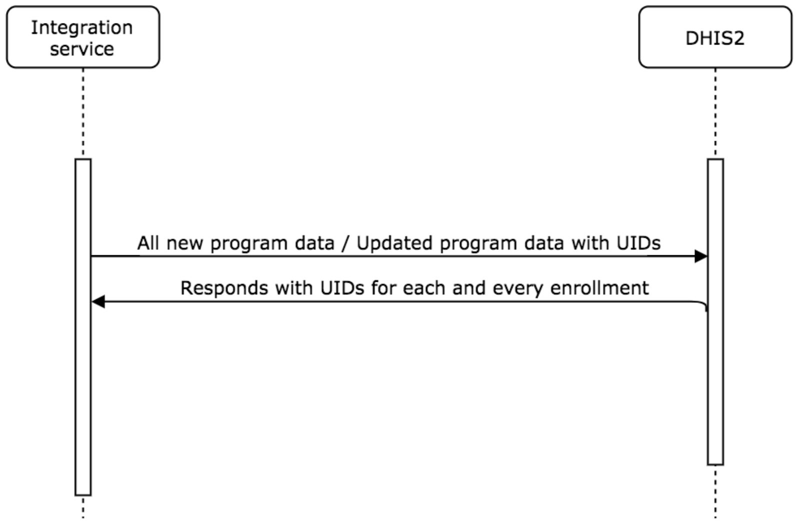
```
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 0,
        "updated": 1,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {
            "idSchemes": {},
            "dryRun": false,
            "async": false,
            "importStrategy": "CREATE_AND_UPDATE",
            "mergeMode": "REPLACE",
            "reportMode": "FULL",
            "skipExistingCheck": false,
            "sharing": false,
            "skipNotifications": false,
            "datasetAllowsPeriods": false,
            "strictPeriods": false,
            "strictCategoryOptionCombos": false,
            "strictAttributeOptionCombos": false,
            "strictOrganisationUnits": false,
            "requireCategoryOptionCombo": false,
            "requireAttributeOptionCombo": false,
            "skipPatternValidation": false
        },
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {
                    "idSchemes": {},
                    "dryRun": false,
                    "async": false,
                    "importStrategy": "CREATE_AND_UPDATE",
                    "mergeMode": "REPLACE",
                    "reportMode": "FULL",
                    "skipExistingCheck": false,
                    "sharing": false,
                    "skipNotifications": false,
                    "datasetAllowsPeriods": false,
                    "strictPeriods": false,
                    "strictCategoryOptionCombos": false,
                    "strictAttributeOptionCombos": false,
                    "strictOrganisationUnits": false,
                    "requireCategoryOptionCombo": false,
                    "requireAttributeOptionCombo": false,
                    "skipPatternValidation": false
                },
                "importCount": {
                    "imported": 0,
                    "updated": 1,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "cqrSZzflgkO",
                "href": "http://192.168.33.23:8080/api/trackedEntityInstances/cqrSZzflgkO",
                "relationships": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "total": 0
                },
                "enrollments": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "total": 0
                }
            }
        ],
        "total": 1
    }
}
```

3.3.2 Sync Newly Completed Enrollments with its Events

For enrolling patients into a program one will need the identifier of the patient (which was generated as UID in the previous step from the *trackedEntityInstances* resource). For enrolling a new patient into a program, you will be working with the *programs* resource.
<u>NEW Patient enrollment:</u> Every successful enrollment would be associated with a unique program identifier UID in DHIS2.
<u>UPDATE Patient enrollment</u>: For updating any enrollment the same UID must be sent as “enrollment” to update its details in DHIS2.
Note: Deletion of enrollment is out of scope.

Observations captured as part of a program in Bahmni world is analogous to the event data in DHIS2 world.
For capturing event for a program one will need the unique id of the patient (which was generated as UID in the previous step from the programs resource) which is mandatory for events with registration. For capturing a new event for a program, you will be working with the programs resource, programStages resource, trackedEntityInstances resource and organisationUnits resource of DHIS2.

<u>NEW event</u>: Every successful event would be associated with a unique event identifier UID in DHIS2. The number of successful data values imported would be mentioned in the response as imported count.
<u>UPDATE event</u>: For updating any event the same UID must be sent as “event” to update its details in DHIS2. However the import count in updation would represent the updates for number of patients along with 200 success code would be received.
Note: All the events will be synced with status COMPLETED.


Integration service
DHIS2
All new program data / Updated program data with UIDs
Responds with UIDs for each and every enrollment

URL: http://<DHIS2-SERVER>/api/enrollments?strategy=CREATE_AND_UPDATE

Request Body (with ACTIVE status) Sample:

```
{
  "enrollments": [
    {
      "trackedEntityInstance": "pWjVB3Pc995",
      "enrollment": "",
      "orgUnit": "kk3nmB4K8ko",
      "program": "SsrIUFxWi5N",
      "enrollmentDate": "2018-12-24",
      "incidentDate": "2018-12-23",
      "status": "ACTIVE",
      "events": [
        {
          "event": "",
          "trackedEntityInstance": "pWjVB3Pc995",
          "orgUnit": "kk3nmB4K8ko",
          "program": "SsrIUFxWi5N",
          "programStage": "crB7sI7gXhZ",
          "eventDate": "2018-12-24",
          "status": "COMPLETED",
          "dataValues": [
            {
              "dataElement": "Ym0UD4gkNGF",
              "value": "Always"
            }
          ]
        }
      ]
    }
  ]
}
```

Response Body (Success) Sample:

```json
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 1,
        "updated": 0,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {},
                "importCount": {
                    "imported": 1,
                    "updated": 0,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "KErPwMosFu0",
                "href": "http://192.168.33.21:8080/api/enrollments/KErPwMosFu0",
                "events": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 1,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "importSummaries": [
                        {
                            "responseType": "ImportSummary",
                            "status": "SUCCESS",
                            "importCount": {
                                "imported": 1,
                                "updated": 0,
                                "ignored": 0,
                                "deleted": 0
                            },
                            "reference": "nu35IMefGcW"
                        }
                    ],
                    "total": 1
                }
            }
        ],
        "total": 1
    }
}

```

Response Body (Failure) Sample:

```json
{
    "httpStatus": "Conflict",
    "httpStatusCode": 409,
    "status": "ERROR",
    "message": "Program has another active enrollment going on. Not possible to incomplete"
}
```

Request Body (with COMPLETED status) Sample:

```
{
  "enrollments": [
    {
      "trackedEntityInstance": "pWjVB3Pc995",
      "enrollment": "KErPwMosFu0",
      "orgUnit": "kk3nmB4K8ko",
      "program": "SsrIUFxWi5N",
      "enrollmentDate": "2018-12-24",
      "incidentDate": "2018-12-23",
      "status": "COMPLETED"
    }
  ]
}

```

```
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 0,
        "updated": 1,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {},
                "importCount": {
                    "imported": 0,
                    "updated": 1,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "KErPwMosFu0",
                "href": "http://192.168.33.21:8080/api/enrollments/KErPwMosFu0",
                "events": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "total": 0
                }
            }
        ],
        "total": 1
    }
}

```

3.3.3 Sync Updated Completed Enrollments with its Events

The URL and API format remains the same as above.

Request Body Sample

```json
{
  "enrollments": [
    {
      "trackedEntityInstance": "pWjVB3Pc995",
      "enrollment": "KErPwMosFu0",
      "orgUnit": "kk3nmB4K8ko",
      "program": "SsrIUFxWi5N",
      "enrollmentDate": "2018-12-24",
      "incidentDate": "2018-12-23",
      "status": "COMPLETED"
    }
  ]
}

```

Response Body (Success) Sample:

```json
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 0,
        "updated": 1,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {},
                "importCount": {
                    "imported": 0,
                    "updated": 1,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "KZpLJCwfOir",
                "href": "http://192.168.33.21:8080/api/enrollments/KZpLJCwfOir",
                "events": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 1,
                    "deleted": 0,
                    "ignored": 0,
                    "importSummaries": [
                        {
                            "responseType": "ImportSummary",
                            "status": "SUCCESS",
                            "importCount": {
                                "imported": 0,
                                "updated": 1,
                                "ignored": 0,
                                "deleted": 0
                            },
                            "reference": "eoKoa0r49lM"
                        }
                    ],
                    "total": 1
                }
            }
        ],
        "total": 1
    }
}
```

3.3.4 Sync New Active Enrollments with its Events

Request Body Sample:

```json
{
  "enrollments": [
    {
      "trackedEntityInstance": "pWjVB3Pc995",
      "enrollment": "",
      "orgUnit": "kk3nmB4K8ko",
      "program": "SsrIUFxWi5N",
      "enrollmentDate": "2018-12-24",
      "incidentDate": "2018-12-23",
      "status": "ACTIVE",
      "events": [
        {
          "event": "",
          "trackedEntityInstance": "pWjVB3Pc995",
          "orgUnit": "kk3nmB4K8ko",
          "program": "SsrIUFxWi5N",
          "programStage": "crB7sI7gXhZ",
          "eventDate": "2018-12-24",
          "status": "COMPLETED",
          "dataValues": [
            {
              "dataElement": "Ym0UD4gkNGF",
              "value": "Always"
            }
          ]
        }
      ]
    }
  ]
}
```

Response Body (Success) Sample:

```
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 1,
        "updated": 0,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {},
                "importCount": {
                    "imported": 1,
                    "updated": 0,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "KErPwMosFu0",
                "href": "http://192.168.33.21:8080/api/enrollments/KErPwMosFu0",
                "events": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 1,
                    "updated": 0,
                    "deleted": 0,
                    "ignored": 0,
                    "importSummaries": [
                        {
                            "responseType": "ImportSummary",
                            "status": "SUCCESS",
                            "importCount": {
                                "imported": 1,
                                "updated": 0,
                                "ignored": 0,
                                "deleted": 0
                            },
                            "reference": "nu35IMefGcW"
                        }
                    ],
                    "total": 1
                }
            }
        ],
        "total": 1
    }
}
```

Response Body (Failure) Sample:

```json
{
    "httpStatus": "Conflict",
    "httpStatusCode": 409,
    "status": "ERROR",
    "message": "An error occurred, please check import summary.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "ERROR",
        "imported": 0,
        "updated": 0,
        "deleted": 0,
        "ignored": 1,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "ERROR",
                "importOptions": {},
                "description": "TrackedEntityInstance pWjVB3Pc995 already has an active enrollment in program SsrIUFxW
                "importCount": {
                    "imported": 0,
                    "updated": 0,
                    "ignored": 1,
                    "deleted": 0
                }
            }
        ],
        "total": 1
    }
}
```

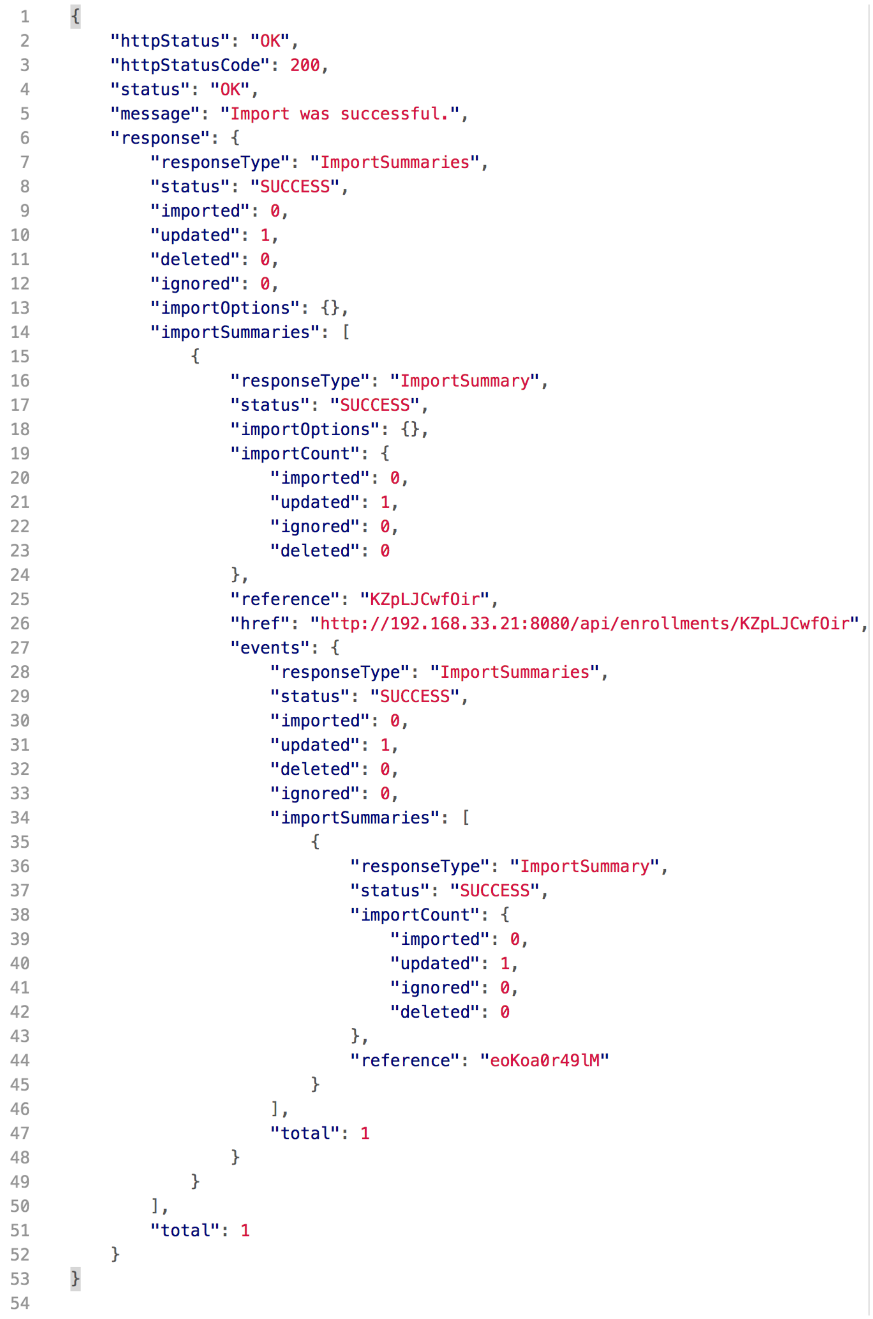
3.3.5 Sync Updated Active Enrollments with its Events

Request Body Sample:

```json
{
  "enrollments": [
    {
      "trackedEntityInstance": "pWjVB3Pc995",
      "enrollment": "KErPwMosFu0",
      "orgUnit": "kk3nmB4K8ko",
      "program": "SsrIUFxWi5N",
      "enrollmentDate": "2018-12-24",
      "incidentDate": "2018-12-23",
      "status": "ACTIVE",
      "events": [
        {
          "event": "",
          "trackedEntityInstance": "pWjVB3Pc995",
          "orgUnit": "kk3nmB4K8ko",
          "program": "SsrIUFxWi5N",
          "programStage": "crB7sI7gXhZ",
          "eventDate": "2018-12-24",
          "status": "COMPLETED",
          "dataValues": [
            {
              "dataElement": "Ym0UD4gkNGF",
              "value": "Always"
            }
          ]
        }
      ]
    }
  ]
}

```

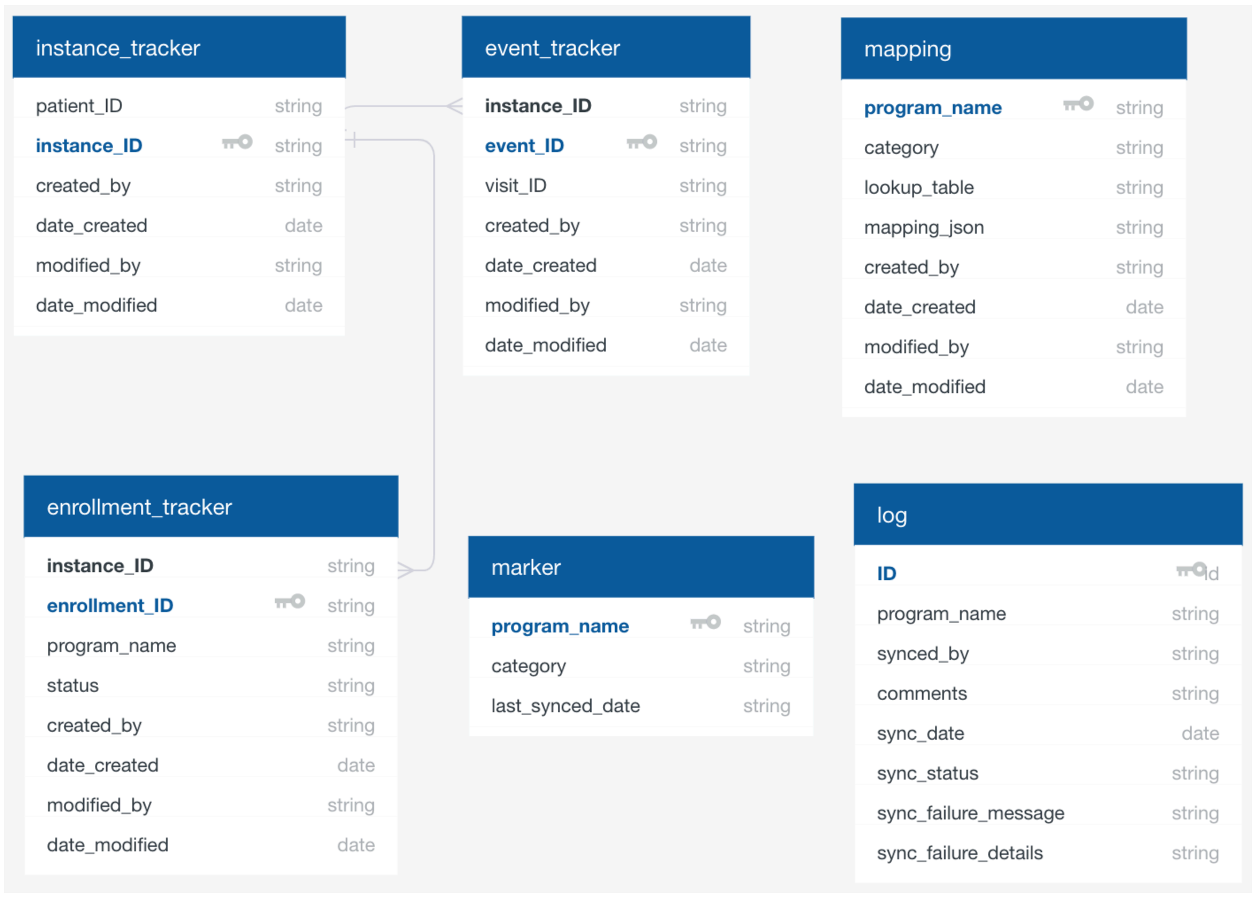
Response Body (Success) Sample:



```
{
    "httpStatus": "OK",
    "httpStatusCode": 200,
    "status": "OK",
    "message": "Import was successful.",
    "response": {
        "responseType": "ImportSummaries",
        "status": "SUCCESS",
        "imported": 0,
        "updated": 1,
        "deleted": 0,
        "ignored": 0,
        "importOptions": {},
        "importSummaries": [
            {
                "responseType": "ImportSummary",
                "status": "SUCCESS",
                "importOptions": {},
                "importCount": {
                    "imported": 0,
                    "updated": 1,
                    "ignored": 0,
                    "deleted": 0
                },
                "reference": "KZpLJCwfOir",
                "href": "http://192.168.33.21:8080/api/enrollments/KZpLJCwfOir",
                "events": {
                    "responseType": "ImportSummaries",
                    "status": "SUCCESS",
                    "imported": 0,
                    "updated": 1,
                    "deleted": 0,
                    "ignored": 0,
                    "importSummaries": [
                        {
                            "responseType": "ImportSummary",
                            "status": "SUCCESS",
                            "importCount": {
                                "imported": 0,
                                "updated": 1,
                                "ignored": 0,
                                "deleted": 0
                            },
                            "reference": "eoKoa0r49lM"
                        }
                    ],
                    "total": 1
                }
            }
        ],
        "total": 1
    }
}
```

3.4 Data Model

[edit the model](#)



instance_tracker	
patient_ID	string
instance_ID	string
created_by	string
date_created	date
modified_by	string
date_modified	date

event_tracker	
instance_ID	string
event_ID	string
visit_ID	string
created_by	string
date_created	date
modified_by	string
date_modified	date

mapping	
program_name	string
category	string
lookup_table	string
mapping_json	string
created_by	string
date_created	date
modified_by	string
date_modified	date

enrollment_tracker	
instance_ID	string
enrollment_ID	string
program_name	string
status	string
created_by	string
date_created	date
modified_by	string
date_modified	date

marker	
program_name	string
category	string
last_synced_date	string

log	
ID	id
program_name	string
synced_by	string
comments	string
sync_date	date
sync_status	string
sync_failure_message	string
sync_failure_details	string

To translate the data from Bahmni world to DHIS2 world the Integration Service would maintain a database with the following tables:

1) Mapping Table
2) Tracker Tables
3) Marker Table
4) Log Table

3.4.1 Mapping table

mapping_name	lookup_table	mapping_json	config
HTS	{"instance":"hts_instance_table","enrollments":"hts_program_enrollment_view","event":"hts_program_events_table"}	{"instance":{"Name":"asdf","Age":"dfgdfg", “Gender”:”jfore73dg”},"event":{"phtc":"asdf"}}	{"searchable":["Name"],"comparable":["Age", “Gender”],"openLatestCompletedEnrollment":"yes"}
TBS	{"instance":"tbs_instance_table","enrollments":"tbs_program_enrollment_view","event":"tbs_program_events_table"}	{"instance":{"Name":"asdf","Age":"dfgdfg", “Gender”:”jfore73dg”},"event":{"fever":"asdf"}}	{"searchable":[“Name”],"comparable":[],"openLatestCompletedEnrollment":"yes"}
FPS	{"instance":"fps_instance_table","enrollments":"fps_program_enrollment_view","event":"fps_program_events_table"}	{"instance":{"Name":"asdf","Age":"dfgdfg", “Gender”:”jfore73dg”},"event":{"semester":"asdf"}}	{"searchable":[],"comparable":[],"openLatestCompletedEnrollment":"no"}

3.4.2 Tracker tables

The Tracker table is normalised into 3 tables

A. Instance_tracker table
B. Enrollment_tracker table
C. Events_tracker table

A. Instance_tracker table

patient_ID	**instance_ID**
NAH009786	CHhFLGOCnza
NAH009624	MkWlpA4hdgY

B. Enrollment_tracker table

instance_ID	**enrollment_ID**	**program**	**program_unique_id**	**status**
CHhFLGOCnza	Y4Z9kEMOnGD	SsrIUFxWi5N	34	ACTIVE
MkWlpA4hdgY	ud1D1IlngW8	qjiYT75fJEPQ	55	ACTIVE

C. Event_tracker table

instance_ID	**event_ID**	**event_unique_id**	**program**	**program_stage**
CHhFLGOCnza	qT40zapm99o	1049	SsrIUFxWi5N	huegsu65KD
MkWlpA4hdgY	tYcCYkLXX9Q	1048	qjiYT75fJEPQ	kdYEP75jdJ

3.4.3 Marker table

program_name	category	last_synced_date
HTS	Instance	2018-06-17 20:30:00.0
TBS	Enrollment	2018-04-11 22:00:00.0
HTS	Enrollment	2018-06-17 20:30:00.0
HTS	Event	2018-06-17 20:30:00.0

3.4.4 Log table

audit_log_ID	program_name	synced_by	comments	sync_date	sync_status	sync_failure_message
1	HTS	admin	Validated on 22 Feb	2018-02-22	Success	
2	TBS	admin	Validated on 23 Feb	2018-02-23	Failed	Please contact Admin team.

Appendix

The supported enrollment sync scenarios for the config “Do you want to open latest completed enrollment?” is set to YES.

New Completed		
Bahmni	**DHIS**	**Final Status**
Completed	Active UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID1
Completed	Completed UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID 1
Completed	Completed UID 1 Completed UID 2	DHIS - Completed UID1 DHIS - Completed UID 2 Bahmni Tracker - Completed UID 2

Update Completed		
Bahmni	**DHIS**	**Final Status**
Completed UID 1	Active UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID1
Completed UID 1	Completed UID 1 Active UID 2	Confict
Completed UID 1	Completed UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID1
Completed UID 1	Completed UID 1 Completed UID (N)	DHIS - Completed UID1 Bahmni Tracker - Completed UID1

New Active enrolment		
Bahmni	**DHIS**	**Final Status**
Active	-	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active	Completed UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active	Active UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1

Update Active		
Bahmni	**DHIS**	**Final Status**
Active UID 1	Active UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active UID 1	Completed UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active UID 1	Completed UID 1 Active UID 2	Confict

The supported enrollment sync scenarios for the config “Do you want to open latest completed enrollment?” is set to NO.

New Completed		
Bahmni	**DHIS**	**Final Status**
Completed	Active UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID1
Completed	Completed UID 1	DHIS - Completed UID1 DHIS - Completed UID 2 Bahmni Tracker - Completed UID 2
Completed	Completed UID 1 Completed UID 2	DHIS - Completed UID1 DHIS - Completed UID 2 DHIS - Completed UID 3 Bahmni Tracker - Completed UID 3

Update Completed		
Bahmni	**DHIS**	**Final Status**
Completed UID 1	Active UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID1
Completed UID 1	Completed UID 1 Active UID 2	Conflict
Completed UID 1	Completed UID 1	DHIS - Completed UID1 Bahmni Tracker - Completed UID 1
Completed UID 1	Completed UID 1 Completed UID (N)	DHIS - Completed UID1 DHIS - Completed UID (N) Bahmni Tracker - Completed UID 1

New Active		
Bahmni	**DHIS**	**Final Status**
Active	-	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active	Completed UID 1	DHIS - Completed UID1 Bahmni Tracker - Active UID 2
Active 25th Dec	Active UID 1 (23rd Dec)	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active 23rd Dec	Active UID 1 (25th Dec)	DHIS - Active UID1 Bahmni Tracker - Active UID1

Update Active		
Bahmni	**DHIS**	**Final Status**
Active UID 1	Active UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active UID 1	Completed UID 1	DHIS - Active UID1 Bahmni Tracker - Active UID1
Active UID 1	Completed UID 1 Active UID 2	Confict