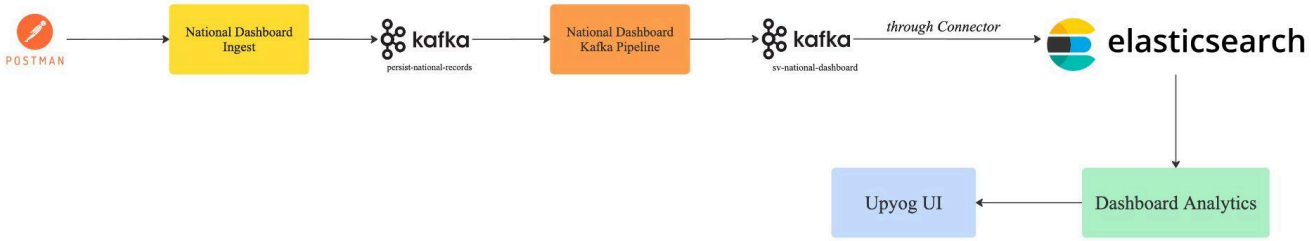
National Dashboard New Module Integration

By - Shivank Shukla

National Dashboard Data Flow Chart



As you see the flow chart you will get how the data is flowing from postman to User interface. Now in this document I will tell you how you can integrate/add a new module in the National Dashboard**.**

To explain more deeply here I am going to take an example of one of my modules - **Street Vending.**

Now we have to run dashboard locally for which we need to run frontend & following microservices:-

1. National Dashboard Ingest
2. National Dashboard Kafka-pipeline
3. Dashboard Analytics
4. Mdms Service

Apart from this we need to run kafka, elasticsearch and kibana as well. Sharing a Doc link so that you can run your dashboard without any issue. How to Run Dashboard locally.

Now when your frontend run successfully do this changes in this directory :-
web/micro-ui-internals/packages/libraries/src/services/atoms/urls.js

Go to this file and do this changes -

```
dss: {
  dashboardConfig: "http://localhost:8289/dashboard-analytics/dashboard/getDashboardConfig",
  getCharts: "http://localhost:8289/dashboard-analytics/dashboard/getChartV2",
 },
```

I have added the localhost path and port as well because my dashboard analytics running locally, port may be vary so change accordingly. I am doing this because when I run my national dashboard from the frontend, all my requests will hit this port/url so that we can easily make changes locally.

I assume now that all of the services are running locally to make this all work. So first thing first we need to finalize KPIs of the module.

1. Changes in National Dashboard Ingest Service.

We start by adding the name of the index in the application.properties, under the index configuration add your index name in ***module.index.mapping,*** take the reference of other indexes as well. When the national dashboard ingest metrics API is hit through postman, all the data payload lookup keys are first checked against the module field mapping, module allowed group by mapping to determine whether they already exist or not. Inside ***module-fields-mapping*** you can configure your KPIs as well as if you are sending any array of objects inside the postman payload then you have to add that inside ***module-allowed-groupby-fields-mapping.***
Now if you have done everything correctly so you can try hitting the data payload from the postman, make sure the port is same as the port on which ***ingest service*** is running and if everything is correct it will save your data in this table - ***nss-ingest-keydata*** with a responseHash.
So once you hit the data from postman, the data will come in ingest service, it will validate the data and if all seems correct it will push the data in ***persist-national-record*** topic and our next service ***National Dashboard Kafka Pipeline*** will listen and consume the record from this ***persist-national-record*** topic.

2. Changes in National Dashboard Kafka Pipeline

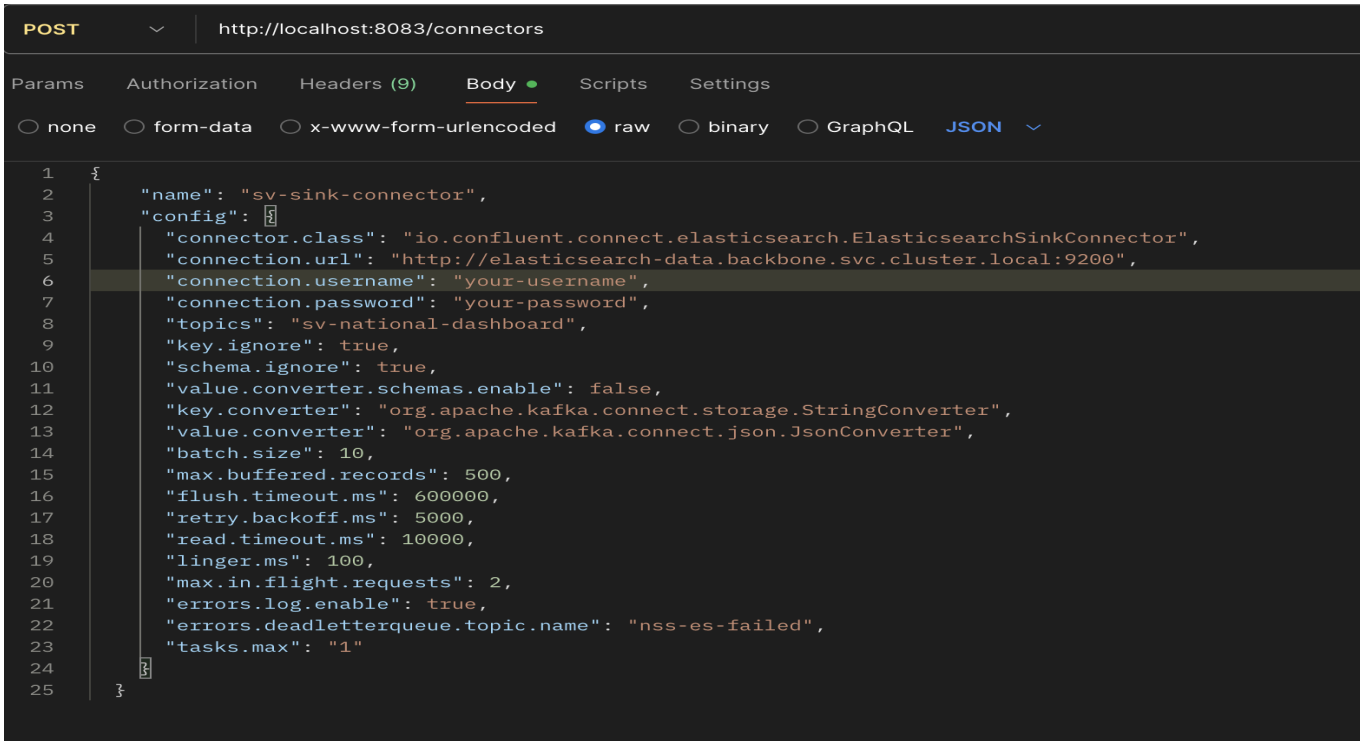
Eventually, you won't need to make any code changes in the National Dashboard Kafka pipeline, except in the ***application.properties*** file, inside ***application.properties*** you have to mention your index name under index configuration, index name should be same as you have mentioned in ***ingest service*** moule-index-mapping.

Now if everything is correct, ***National Dashboard Kafka Pipeline*** will listen and consume the record from ***persist-national-record*** topic and push the data in your topic for example push the data in ***sv-national-dashboard*** which we added in module-index-mapping and from this topic, data will flow to ***elasticsearch*** through connector.

3. How to Develop Connector

Why Connector? We use a ***Connector*** to automatically transfer data from Kafka topics to ***Elasticsearch*** without writing custom code. It consumes messages from Kafka and indexes them into Elasticsearch in near real-time. Built on top of Kafka Connect, it's fault-tolerant, supports offsets, retries, and scaling with multiple tasks.

Adding a screenshot of the curl for your understanding and sharing a curl as well



As you can see in the screenshot you just have to change the ***name*** here, add a unique name as well as a ***topic*** for this connector as well for example i have added ***sv-national-dashboard*** name because this is the name i configured in my ***Ingest service*** and ***National Dashboard Kafka Pipeline.*** Make sure to enter the correct username and password of the elasticsearch in connection.username and connection.password. Sharing the curl for your reference below.

```
curl --location 'http://localhost:8083/connectors' \
--header 'Content-Type: application/json' \
--data '{
    "name": "sv-sink-connector",
    "config": {
      "connector.class": "io.confluent.connect.elasticsearch.ElasticsearchSinkConnector",
      "connection.url": "http://elasticsearch-data.backbone.svc.cluster.local:9200",
      "connection.username": "your-username",
      "connection.password": "your-password",
      "topics": "sv-national-dashboard",
      "key.ignore": true,
      "schema.ignore": true,
      "value.converter.schemas.enable": false,
      "key.converter": "org.apache.kafka.connect.storage.StringConverter",
      "value.converter": "org.apache.kafka.connect.json.JsonConverter",
      "batch.size": 10,
      "max.buffered.records": 500,
      "flush.timeout.ms": 600000,
```

```
    "retry.backoff.ms": 5000,
    "read.timeout.ms": 10000,
    "linger.ms": 100,
    "max.in.flight.requests": 2,
    "errors.log.enable": true,
    "errors.deadletterqueue.topic.name": "nss-es-failed",
    "tasks.max": "1"
  }
 }'
```

Now to execute the connector, go to your kubeconfig and check the pod name - kafka-connect
And copy it, now inside your kubeconfig run this command -
kubectl exec -it ***name of your kafka-connect pod*** *-n kafka-kraft -- /bin/sh*
Once run above command, paste your curl of connector and hit enter and that's how you successfully configured your connector, to check run this command after that -
curl -X GET http://localhost:8083/connectors it will show all the connectors which you have.

There is one more way to make connectors but that will only be applicable for your local use only for your production environment you have to make connectors like this only. For local use I have already shared the doc in which I have mentioned how to run the dashboard locally including elasticsearch and kibana with error resolution steps.

Now if you have done everything right your pushed data will successfully flow to the elastic search and you can check that data in Kibana.

Now open your ***kibana,*** and start putting your indexes, for that you need to login inside your kibana which you will get when you run elasticsearch on your local terminal after login go to ***dev-tools*** from sidebar and make a PUT query to put index in elasticsearch inside these PUT all the keys should match to your postman payload as well as module-fields-mapping as well. For example sharing you street-vending one -

```
PUT /sv-national-dashboard
{
  "mappings": {
    "properties": {
      "date": {
        "type": "date",
        "format": "dd-MM-yyyy||epoch_millis"
      },
      "module": { "type": "keyword" },
      "region": { "type": "keyword" },
      "state": { "type": "keyword" },
      "ulb": { "type": "keyword" },
      "ward": { "type": "keyword" },
```

```
    "createdBy": { "type": "keyword" },
    "createdTime": { "type": "long" },
    "lastModifiedBy": { "type": "keyword" },
    "lastModifiedTime": { "type": "long" },
    "timestamp": {"type": "date"},
    "totalUrbanHouseholds": { "type": "integer" },
    "totalStreetVendingLicenseApplied": { "type": "integer" },
    "percentageLicenseIssuedVsApplied": { "type": "float" },
    "averageTATLicenseIssued": { "type": "integer" },
    "vendingType": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    },
    "totalLicenseAppliedbyMale": { "type": "integer" },
    "totalLicenseAppliedbyFemale": { "type": "integer" },
    "licenseByAge_60plus": { "type": "integer" },
    "licenseByAge_30_45": { "type": "integer" },
    "fixedSpotsOfferedByStateUT": { "type": "integer" },
    "licenseByAge_18_30": { "type": "integer" },
    "numberOfApplicationsRejected": { "type": "integer" },
    "rejectionRate": { "type": "float" },
    "numberOfVendingZones": { "type": "integer" },
    "totalStreetVendingLicenseIssued": { "type": "integer" },
    "typeOfRegistration": { "type": "integer" },
    "fixedSpotsOfferedByULB": { "type": "integer" },
    "totalRevenueCollected": { "type": "long" },
    "totalStreetVendors": { "type": "integer" },
    "totalRevenueTargeted": { "type": "long" },
    "targetAchievementPercentage": { "type": "float" },
    "licenseByAge_45_60": { "type": "integer" }
   }
  }
}
```

This put query will hit to create the index, now when you hit the data from postman by running put you will able to see the data you pushed in ***Kibana.***
To check the pushed data in kibana use this query -

```
GET sv-national-dashboard/_search
{
  "query": {
    "match_all": {}
  },
  "size": 10000
}
```

When you run this Get query you will able to see your pushed data if you have do everything as suggested in this Doc.

4. Changes in Dashboard Analytics

Now when you open your dashboard in UI, all of your api calls are hitting the port in which your dashboard analytics is running. If not then please do the changes in urls.js.

Now go to the dashboard-analytics code → src→main→resources→schema here you will find 3 major files, name for the same given below:-

- Chart API Configuration
- Master Dashboard Configuration
- Role Dashboard Mappings Configuration

Update these files as per your dashboard analytics in your Upyog-config Git Repository, directory is ***configs/egov-dss-dashboards/dashboard-analytics,*** you will find same files here so just the code from here and update in dashboard analytics code it will help you to get synced up.

Go to Role Dashboard Mappings Configuration code and add your new module configurations here for example - {

```
 "name": "National Street Vending",
 "id": "national-sv"
}
```

Now inside your Chart API Configuration you have to write all of your aggregation query of Elasticsearch for example -

```
"svTotalStreetVendors": {
 "chartName": "DSS_SV_TOTAL_STREET_VENDORS",
 "queries": [
   {
     "module": "SV",
     "indexName": "sv-national-dashboard",
                "requestQueryMap":    "{\"wardId\":\"ward\",    \"district\":\"district\",
\"tenantId\":\"ulb\"}",
     "dateRefField": "date",
         "aggrQuery":  "{\"aggs\":{\"AGGR\":{\"filter\":{\"bool\":{}},\"aggs\":{\"Total
Vendors\":{\"sum\":{\"field\":\"totalStreetVendors\"}}}}}}"
   }
 ],
 "chartType": "metric",
 "valueType": "number",
 "drillChart": "none",
 "documentType": "_doc",
```

```
 "action": "",
 "aggregationPaths": ["Total Vendors"],
 "insight": {},
 "_comment": "SV total registered vendors"
},
```

As you see above I made a query for the svTotalStreetVendors, which will return total street vendors. Inside the query I have written the module name, index name as well as aggregation query as well.

So if you have to make any new query then you have to make it like this, also after making the aggregation query please convert the aggregation query so that you can run in kibana. Only then add your aggregation query here.

Chart API Configuration - Each Visualization has its own properties. Each Visualization comes from different data sources (Sometimes it is a combination of different data sources).

Now go to this file Master Dashboard Configuration and add your config here like this

```
{
 "id": "svTotalStreetVendors",
 "name": "NSS_SV_TOTAL_VENDORS",
 "code": "",
 "chartType": "metric",
 "filter": "",
 "headers": []
},
```

Here you can see the id is the same as i made in the chart api config svTotalStreetVendors.

In the same way, you can make all of your visualizations.

FINANCE_DASHBOARD

https://docs.google.com/document/d/1QqppJ1XWmXQEPzQ5dFJEuv-dzgCIdaqJuLlIFYST5tg/edit?usp=sharing