Ministry of Housing and Urban Affairs
Government of India

Ministry of Housing and Urban Affairs
Government of India

National Urban Digital Mission

Building cities that work for people

Report Service and PDF Generator

Outline

1. Reporting Service and its usage
2. Which types of report should report service be used for
3. Impact of heavy reports on the platform
4. Configuring a new report
5. Configuring PDF generator

Reporting Service and its usage

Reporting Service is a service running independently on separate server. The main objective of this service is to provide a common framework for generating reports. This service loads the report configuration from a yaml file at the run time and provides the report details by using couple of APIs.

https://upyog.niua.org/employee/report/pt-reports/PTReceiptRegister

https://github.com/nugp-digit/nugp-configs/blob/master/configs/reports/config/pt-reports.yml

- **Configuration**: As mentioned above, report service uses a config file per module to store all the configurations of reports pertaining to that module. Report service reads multiple such files at start-up to support reports of all the configured modules. The file contains the following keys:
  - **reportName:** name of report, to be used with module name to identify any report config
- **summary:** summary of report
- **version:** version of report
- **moduleName:** name of module which the report belongs to

- **sourceColumns :** These represent the final data sent by service on GET_DATA api call. Order of sourceColumns in the Config is same as that of columns in the result. Each sourceColumns represent one column in the result. For each column data is picked after executing final SQL query formed after appending groupby, orderby, search params into base query
    - **name**: name of column to fetch data from query results, must be there in query results
    - **label**: custom column label
    - **type**: data type of column
    - **source**: module name
    - **total**: whether column total required on front end
- **searchParams**:
    - **name**: name of search param. Must match variable used in search clause
    - **label**: custom label for viewing on front end
    - **type**: type of search params. If type is ‘singlevaluelist’ then use pattern to populate searchparams possible values to select from by the user
      Ex:-number,string,singlevaluelist etc
    - **source**: module name
    - **isMandatory**: If user must fill this searchparam before requesting report data

- **searchClause:** SQL search clause for corresponding search params to filter results, to be appended in base query Ex:- AND fnoc.tenantId IN ($ulb)
  Here $ulb will be replaced by user inputs
- **Pattern:** This field will be used only when ‘type’ is set to ‘singlevaluelist’. It is external service URL combined with JSON Paths separated by ‘|’. The first JSON path is for codes and second for values. Values will be shown to the user in drop down. And codes corresponding to user selected value will be sent to the report service and will be used in searchClauses mentioned in last point.
- **Query:** Main/base query clause for fetching report data from DB and custom tables formed after fetching data from external service
- **Orderby:** order by clause to be appended into base query
- **Groupby:** group by clause to be appended into base query
- **additionalConfig:** to provide additional custom configs which are not present above

API Details

There are two API calls to report service ‘GET METADATA’ and ‘GET DATA’.

GET METADATA:

This request to report service is made to get metadata for any report. The metadata contains information about search filters to be used in the report before actually sending request to get actual data. The user selected values are then used in GET_DATA request to filter data.

endpoint: /report/{moduleName}/{report name}/metadata/_get

moduleName:- It is used to define the names of module which contains current report

Body: Body consists the following:

1. RequestInfo: Header details as used on the egov platform
2. tenantId: tenantId of ULB
3. reportName: name of the report to be used

Confuguring a new report

1. Write configuration as per your requirement. Structure of the config file is explained under Configuration Details section
2. Check-in the config file to a remote location preferably github, currently we check the files into this folder - https://github.com/UPYOG-UPYOG/UPYOG-configs/blob/master/configs/reports/config/pt-reports.yml Can't find link for dev and QA environement.
3. Add module name and corresponding report path in same format as used in https://github.com/UPYOG-UPYOG/UPYOG-configs/blob/master/configs/reports/reportFileLocationsv1.txt
4. Provide the absolute path of the file mentioned in Point 3 to DevOps, to add it to the file-read path of report service. The file will be added to environment manifest file for it to be read at start-up of the application.
5. Deploy the latest version of report service app.
6. Add Role-Action mapping for API's.
7. Use the module-name as path parameters in the URL of the requests for report service with required request body.

Ministry of Housing and Urban Affairs
Government of India

PDF-Service

The objective of PDF generation service is to bulk generate pdf as per requirement.

Key Functionalities

- Provide common framework to generate PDF.
- Provide flexibility to customise the PDF as per the requirement.
- Provide functionality to add an image in PDF.
- Provide functionality to generate pdf in bulk.
- Provide functionality to specify maximum number of records to be written in one PDF.

Environment Variables	Description
MAX_NUMBER_PAGES	Maximum number of records to be written in one PDF
DATE_TIMEZONE	Date timezone which will be use to convert epoch timestamp into date (DD/MM/YYYY)
DEFAULT_LOCALISATION_LOCALE	Default value of localisation locale
DEFAULT_LOCALISATION_TENANT	Default value of localisation tenant
DATA_CONFIG_URLS	File path/URL'S of data config
FORMAT_CONFIG_URLS	File path/URL'S of format config

Configuring PDF

https://github.com/UPYOG-UPYOG/UPYOG-configs/tree/master/configs/pdf-service/data-config

https://github.com/UPYOG-UPYOG/UPYOG-configs/tree/master/configs/pdf-service/format-config

1. Create data config and format config for a PDF according to product requirement.
2. Add data config and format config files in PDF configuration
3. Add the file path of data and format config in environment yml file
4. Deploy latest version of pdf-service in particular environment.

The data config file contains the following aspects :

- **key:** The key for the pdf, it is used as a path parameter in URL to identify for which PDF has to generate.
- **baseKeyPath:** The json path for the array object that we need to process.
- **entityIdPath:** The json path for the unique field which is stored in DB. And that unique field value is mapped to file-store id, so we can directly search the pdf which was created earlier with the unique field value and there will be no need to create PDF again.
- **mapping:** There are three mapping object for variable which are direct mapping, externalApi mapping and derived mapping.
    - **Direct Mapping:** In direction mapping we define the variable whose value can be fetched from the array object which we extracted using baseKeyPath.
    - **ExternalApi Mapping:** We use the externalApi mapping only if there is a need of values from other service response. In externalApi mapping, API endpoint has to be set properly with correct query parameter.
    - **Derived mapping:** In derived mapping, the estimation of variable characterize here is equivalent to esteem which acquired from the arithmetic operation between variable of direct mapping and externalApi mapping.

Thank You

"UPYOGal Transformation is more about humans than UPYOGal"