Sewerage Calculator Service - Technical Document

This is one of the major business logic services which is used for calculation of sewerage charge, generating demand, update existing demand , SMS & email notification to the ULB officials on demand generation and also triggering demands(job scheduler) at some intervals and estimation of water charge(one-time cost) which involves cost like road-cutting charge , form fee , scrutiny fee etc.

Requirements

- Knowledge of Java/J2EE(preferably Java 8 version)
- Knowledge of Spring Boot and spring-boot microservices
- Knowledge of Git or any version control system
- Knowledge of RESTful Web services
- Knowledge of the Lombok library will helpful
- Knowledge of eGov-mdms service, eGov-persister, eGov-idgen, eGov-sms, eGov-email,eGov-user, eGov-localization, eGov-workflow-service will be helpful

Functionality

Sewerage calculator services present in municipal services provides multiple functionalities like calculating sewerage charges, generating demands for a particular sewerage connection , updating demands, SMS & email notification to the ULB officials on demand generation and also triggering demands(job scheduler) at some intervals and estimation of water charge(one-time cost) which involves cost like road-cutting charge , form fee , scrutiny fee etc. The different functionalities provided by sewerage calculator services are:

1. Sewerage charge calculation
2. Demand generation(here as its always non-metered demand will be generated based on time period)

3. Sewerage charge estimation (one-time cost which involves cost like road-cutting charge, form fee, scrutiny fee etc.)

Setup and usage

The **Application** is present among the ***municipal services*** group of applications available in the eGov-services git repository. The spring boot application needs the **Lombok*** extension added in your IDE to load it. Once the application is up and running API requests can be posted to the URL and ids can be generated.

- In case of IntelliJ, the plugin can be installed directly, for eclipse the Lombok jar location has to be added in eclipse.ini file in this format javaagent:lombok.jar

API Information

- Please refer Swagger API for YAML file details. Link - https://app.swaggerhub.com/apis/egov-foundation/Water-Sewerage-1.0/1.0.0.

Application.properties File Information

kafka topics persister configs for eGov persister (for notification)

- kafka.topics.notification.sms=egov.core.notification.sms
- notification.sms.enabled=true
- kafka.topics.notification.mail=notification.mail
- notification.mail.enabled=true
- kafka.topics.notification.mail.name=egov.core.notification.email
- egov.seweragecalculatorservice.createdemand=sw-generate-demand
- notification.sms.link=citizen/egov-common/pay?consumerCode=$consumerCode&tenantId=$tenantId&businessService=SW

URLs for the external API references

- eGvo mdms :-> egov.mdms.host = https://egov-micro-dev.egovernments.org
- eGov -idGen :-> egov.idgen.host = https://egov-micro-dev.egovernments.org/
- localization service :-> egov.localization.host = https://egov-micro-dev.egovernments.org/
- user-service :-> egov.user.host = https://egov-micro-dev.egovernments.org/
- Url-Shortner egov.url.shortner.host=http://egov-url-shortening.egov:8080

Billing Slabs

Criteria

1. connection type
2. building type
3. calculation attribute
4. property usage type

The combination of the above can be used to define the billing slab. Billing Slab is defined in MDMS under sw-services-calculation folder with the SCBillingSlab. The following is the sample slab.

```
{
  "tenantId": "pb",
  "moduleName": "sw-services-calculation",
  "SCBillingSlab": [
    {
      "id": "1",
      "buildingType": "RESIDENTIAL",
      "calculationAttribute": "No. of water closets",
      "connectionType": "Non Metered",
      "minimumCharge": 0,
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 15
        }
      ]
    },
```

```
    {
      "id": "2",
      "buildingType": "RESIDENTIAL",
      "calculationAttribute": "No. of toilets",
      "connectionType": "Non Metered",
      "minimumCharge": 0,
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 15
        }
      ]
    },
    {
      "id": "3",
      "buildingType": "NONRESIDENTIAL",
      "calculationAttribute": "No. of water closets",
      "connectionType": "Non Metered",
      "minimumCharge": 0,
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
    },
    {
      "id": "4",
      "buildingType": "NONRESIDENTIAL",
      "calculationAttribute": "No. of toilets",
      "connectionType": "Non Metered",
      "minimumCharge": 0,
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
    },
    {
      "id": "5",
      "buildingType": "Commercial",
      "calculationAttribute": "No. of water closets",
      "connectionType": "Non Metered",
      "minimumCharge": 0,
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
```

```
    },
    {
      "id": "6",
      "buildingType": "Commercial",
      "calculationAttribute": "No. of toilets",
      "connectionType": "Non Metered",
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
    },
    {
      "id": "7",
      "buildingType": "Government",
      "calculationAttribute": "No. of water closets",
      "connectionType": "Non Metered",
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
    },
    {
      "id": "8",
      "buildingType": "Government",
      "calculationAttribute": "No. of toilets",
      "connectionType": "Non Metered",
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 30
        }
      ]
    },
    {
      "id": "9",
      "buildingType": "Partly Commercial",
      "calculationAttribute": "No. of water closets",
      "connectionType": "Non Metered",
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 25
        }
      ]
    },
    {
      "id": "10",
```

```
      "buildingType": "Partly Commercial",
      "calculationAttribute": "No. of toilets",
      "connectionType": "Non Metered",
      "slabs": [
        {
          "from": 0,
          "to": 1000000000,
          "charge": 25
        }
      ]
    },
    {
      "id": "11",
      "buildingType": "RESIDENTIAL",
      "calculationAttribute": "Flat",
      "connectionType": "Non Metered",
      "minimumCharge": 100,
      "slabs": []
    },
    {
      "id": "12",
      "buildingType": "NONRESIDENTIAL",
      "calculationAttribute": "Flat",
      "connectionType": "Non Metered",
      "minimumCharge": 250,
      "slabs": []
    },
    {
      "id": "13",
      "buildingType": "Commercial",
      "calculationAttribute": "Flat",
      "connectionType": "Non Metered",
      "minimumCharge": 250,
      "slabs": []
    },
    {
      "id": "14",
      "buildingType": "Government",
      "calculationAttribute": "Flat",
      "connectionType": "Non Metered",
      "minimumCharge": 350,
      "slabs": []
    },
    {
      "id": "15",
      "buildingType": "Partly commercial",
      "calculationAttribute": "Flat",
      "connectionType": "Non Metered",
      "minimumCharge": 200,
      "slabs": []
    }
  ]
}
```

If all criteria will match for that sewerage connection this slab will use for calculation.

Estimation

For application one-time fee, the estimation will return all the related tax head based on criteria. For estimation, all configurations are present in ws-services-calculation.

1. FeeSlab.json
2. PlotSizeSlab.json
3. RoadType.json

The above master configurations are used for estimation. Following are the exemptions and taxes that are calculated:

- Form fee
- Scrutiny fee
- Other charges
- Road cutting charges
- One time fee
- Security charges
- Tax and cess

Sewerage Charge and Tax

Sewerage charge is based on billing slab, for sewerage application charge will be based on slab and tax based on master configuration.

Interest

Below is a sample of master data JSON for interest:

```
{
  "tenantId": "pb",
  "moduleName": "sw-services-calculation",
  "Interest": [
    {
      "rate": 5,
```

```
      "minAmount": null,
      "applicableAfterDays":0,
      "flatAmount": null,
      "maxAmount": null,
      "fromFY": "2019-20",
      "startingDay": "1/01/2019"
    }
  ]
}
```

Penalty

Below is a sample of master data JSON for penalty:

```
{
  "tenantId": "pb",
  "moduleName": "sw-services-calculation",
  "Penalty": [
    {
      "rate": 10,
      "minAmount": null,
      "applicableAfterDays": 0,
      "flatAmount": null,
      "fromFY": "2019-20",
      "startingDay": "1/01/2019"
    }
  ]
}
```

Round Off

If the fraction is greater than equal to 0.5 the number is round up else it's round down. eg: 100.4 will be rounded to 100 while 100.6 will be rounded to 101.

Demand Generation

Once sewerage is sent to calculator its tax estimates are calculated. Using this tax head estimates demand details are created. For every tax head, estimate demand generates function will create a corresponding demand detail.

Whenever _calculate API is called demand is first searched based on the connection no or application no and the demand from and to period. If demand already exists the same demand is updated else new demand is

generated with consumer code as connection no or application no and demand from and to a period equal to financial year start and end period.

In case of the update if the tax head estimates change, the difference in amount for that tax head is added as new demand detail. For example, if the initial demand has one demand detail with SEWERAGE_CHARGE equal to 120


```
"demandDetails": [
                {
                    "id": "77ba1e93-a535-409c-b9d1-a312c409bd45",
                    "demandId": "687c3176-305b-461d-9cec-2fa26a30c88f",
                    "taxHeadMasterCode": "SEWERAGE_CHARGE",
                    "taxAmount": 120,
                    "collectionAmount": 120,
                    "additionalDetails": null,
                    "auditDetails": {
                        "createdBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "lastModifiedBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "createdTime": 1583675275873,
                        "lastModifiedTime": 1583675298705
                    },
                    "tenantId": "pb.amritsar"
                }
            ],
```


After updating if the SEWERAGE_CHARGE increases to 150 we add one more demand detail to account for the increased amount. The demand detail will be updated to:


```
"demandDetails": [
                {
                    "id": "77ba1e93-a535-409c-b9d1-a312c409bd45",
                    "demandId": "687c3176-305b-461d-9cec-2fa26a30c88f",
                    "taxHeadMasterCode": "SEWERAGE_CHARGE",
                    "taxAmount": 120,
                    "collectionAmount": 0,
                    "additionalDetails": null,
                    "auditDetails": {
                        "createdBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "lastModifiedBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "createdTime": 1583675275873,
                        "lastModifiedTime": 1583675298705
                    },
                    "tenantId": "pb.amritsar"
                },
                {
                    "id": "0d83f4b0-6442-11ea-bc55-0242ac130003 ",
                    "demandId": "687c3176-305b-461d-9cec-2fa26a30c88f",
                    "taxHeadMasterCode": "SEWERAGE_CHARGE",
                    "taxAmount": 30,
```

```
                    "collectionAmount": 0,
                    "additionalDetails": null,
                    "auditDetails": {
                        "createdBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "lastModifiedBy": "04956309-87cd-4526-b4e6-48123abd4f3d",
                        "createdTime": 1583675275873,
                        "lastModifiedTime": 1583675298705
                    },
                    "tenantId": "pb.amritsar"
                }
            ],
```

RoundOff is bill based i.e every time bill is generated round off is adjusted so that payable amount is the whole number. Individual SW_ROUNDOFF in demand detail can be greater than 0.5 but the sum of all SW_ROUNDOFF will always be less than 0.5.

Scheduler for generating the demand

Description

For generating the demand for non metered connection we have a feature for generating the demand in batch. The scheduler is responsible for generating the demand based on the tenant.

- The scheduler can be hit by scheduler API or we can schedule cron job or we can put config to kubectl which will hit scheduler based on config.
- After the scheduler been hit we will search the list of the tenant (city) present in the database.
- After getting the tenants we will pickup tenant one by one and generate the demand for that tenant.
- We will load the consumer codes for the tenant and push the calculation criteria to Kafka. Calculation criteria contain minimal information (We are not pushing large data to Kafka), calculation criteria contain consumer code and one Boolean variable.
- After pushing the data into Kafka we are consuming the records based on the batch configuration. Ex:-> if the batch configuration is 50 so we will consume the 50 calculation criteria at a time.

- After consuming the record (Calculation criteria) we will process the batch for generating the demand. If the batch is successful so will log the consumer codes which have been processed.
- If some records failed in batch so we will push the batch into dead letter batch topic. From the dead letter batch topic, we will process the batch one by one.
- If the record is successful we will log the consumer code. If the record is failed so we will push the data into a dead letter single topic.
- Dead letter single topic contains information about failure records in Kafka.

Use cases

1. If the same job trigger multiple time what will happen?

   If the same job triggers multiple times we will process again as mentioned above but at the demand level we will check the demand based on consumer code and billing period, If demand already exists then we will update the demand otherwise we will create the demand.

2. Are we maintaining success or failure status anywhere?

   Currently, we are maintaining the status of failed records in Kafka.

Configuration

We need to configure the batch size for Kafka consumer. This configuration is for how much data will be processed at a time.

```
sw.demand.based.batch.size=10
```