Indexer Configuration

Indexer uses a config file per module to store all the configurations pertaining to that module. Indexer reads multiple such files at start-up to support indexing for all the configured modules. In config we define source and, destination elastic search index name, custom mappings for data transformation and mappings for data enrichment. Below is the sample configuration for indexing TL application creation data into elastic search.

```
ServiceMaps:
  serviceName: Trade License
  version: 1.0.0
  mappings:
  - topic: save-tl-tradelicense
    configKey: INDEX
    indexes:
    - name: tlindex-v1
      type: licenses
      id: $.id, $.tenantId
      isBulk: true
      timeStampField: $.auditDetails.createdTime
      jsonPath: $.Licenses
      customJsonMapping:
        indexMapping: {"Data":{"tradelicense":{},"ward":{},"tenantData":{}, "history":{}}}
        fieldMapping:
        - inJsonPath: $
          outJsonPath: $.Data.tradelicense
        externalUriMapping:
        - path: http://egov-location.egov:8080/egov-location/location/v11/boundarys/_search
          queryParam:
hierarchyTypeCode=REVENUE,boundaryType=locality,codes=$.tradeLicenseDetail.address.local
ity.code,tenantId=$.tenantId
          apiRequest:
{"RequestInfo":{"apiId":"org.egov.pt","ver":"1.0","ts":1502890899493,"action":"asd","did":"4354648
646","key":"xyz","msgId":"654654","requesterId":"61","authToken":"d9994555-7656-4a67-ab3a-
a952a0d4dfc8","userInfo":{"id":1,"uuid":"1fec8102-0e02-4d0a-b283-
cd80d5dab067","type":"EMPLOYEE","tenantId":"pb.amritsar","roles":[{"name":"Employee","code":"
EMPLOYEE","tenantId":"pb.amritsar"}]}}}
          uriResponseMapping:
          - inJsonPath: $.TenantBoundary[0].boundary[0]
            outJsonPath: $.Data.ward
        - path: http://egov-workflow-v2.egov:8080/egov-workflow-v2/egov-wf/process/_search
          queryParam: businessIds=$.applicationNumber,history=true,tenantId=$.tenantId
          apiRequest:
{"RequestInfo":{"apiId":"org.egov.pt","ver":"1.0","ts":1502890899493,"action":"asd","did":"4354648
646","key":"xyz","msgId":"654654","requesterId":"61","authToken":"d9994555-7656-4a67-ab3a-
a952a0d4dfc8","userInfo":{"id":1,"uuid":"1fec8102-0e02-4d0a-b283-
```

```
cd80d5dab067","type":"EMPLOYEE","tenantId":"pb.amritsar","roles":[{"name":"Employee","code":"
EMPLOYEE","tenantId":"pb.amritsar"}]}}}
        uriResponseMapping:
        - inJsonPath: $.ProcessInstances
          outJsonPath: $.Data.history
      mdmsMapping:
      - path: http://egov-mdms-service.egov:8080/egov-mdms-service/v1/_search
        moduleName: tenant
        masterName: tenants
        tenantId: pb
        filter: "[?(@.code == $tenant)]"
        filterMapping:
        - variable: $tenant
          valueJsonpath: $.tenantId
        uriResponseMapping:
        - inJsonPath: $.MdmsRes.tenant.tenants
          outJsonPath: $.Data.tenantData
```

The configuration file contains following keys:-

Variable Name	**Description**
Variable Name	**Description**
serviceName	Name of the module to which this configuration belongs.
summary	Summary of the module.
version	Version of the configuration.
mappings	List of definitions within the module. Every definition corresponds to one index requirement. Which means, every object received onto the kafka queue can be used to create multiple indexes, each of these indexes will need configuration, all such configurations belonging to one topic forms one entry in the mappings list. The keys listed henceforth together form one definition and multiple such definitions are part of this mappings key.
topic	Topic on which the data is to be received to activate this particular configuration.
configKey	Key to identify to what type of job is this config for. values: INDEX, REINDEX, LEGACYINDEX. INDEX: LiveIndex, REINDEX: Reindex, LEGACYINDEX: LegacyIndex.
indexes	Key to configure multiple index configurations for the data received on the particular topic. Multiple indexes

	based on different requirement can be created using the same object.
name	Index name on the elasticsearch. (Index will be created if it doesn't exist with this name.)
type	Document type within that index to which the index json has to go. (Elasticsearch uses the structure of index/type/docId to locate any file within index/type with id = docId)
id	Takes comma separated JsonPaths. The JSONPath is applied on the record received on the queue, the values hence obtained are appended and used as id for the record.
isBulk	Boolean key to identify whether the JSON received on the Queue is from a Bulk API. In simple words, whether the JSON contains a list at the top level.
jsonPath	Key to be used in case of indexing a part of the input JSON and in case of indexing a custom json where the values for custom json are to be fetched from this part of the input.
timeStampField	JSONPath of the field in the input which can be used to obtain the timestamp of the input.
fieldsToBeMasked	A list of JSONPaths of the fields of the input to be masked in the index.
customJsonMapping	Key to be used while building an entirely different object using the input JSON on the queue
indexMapping	A skeleton/mapping of the JSON that is to be indexed. Note that, this JSON must always contain a key called "Data" at the top-level and the custom mapping begins within this key. This is only a convention to smoothen dashboarding on Kibana when data from multiple indexes have to be fetched for a single dashboard.
fieldMapping	Contains a list of configurations. Each configuration contains keys to identify the field of the input JSON that has to be mapped to the fields of the index json which is mentioned in the key 'indexMapping' in the config.
inJsonPath	JSONPath of the field from the input.
outJsonPath	JSONPath of the field of the index json.
externalUriMapping	Contains a list of configurations. Each configuration

	contains keys to identify the field of the input JSON that are to be enriched using APIs from the external services. The confiugration for those APIs also is a part of this.
path	URI of the API to be used. (it should be POST/_search API.)
queryParam	Configuration of the query params to be used for the API call. It is a comma seperated key-value pair, where key is the parameter name as per the API contract and value is the JSONPath of the field to be equated against this paramter.
apiRequest	Request Body of the API. (Since we only use _search APIs, it should be only RequestInfo.)
uriResponseMapping	Contains a list of configuration. Each configuration contains two keys: One is a JSONPath to identify the field from response, Second is also a JSONPath to map the response field to a field of the index json mentioned in the key 'indexMapping'.
mdmsMapping	Contains a list of configurations. Each configuration contains keys to identify the field of the input JSON that are to be denormalized using APIs from the mdms service. The confiugration for those mdms APIs also is a part of this.
path	URI of the API to be used. (it should be POST/_search API.)
moduleName	Module Name from MDMS.
masterName	Master Name from MDMS.
tenantId	Tenant id to be used.
filter	Filter to be applied on the data to be fetched.
filterMapping	Maps the field of input json to variables in the filter
variable	Variable in the filter
valueJsonpath	JSONPath of the input to be mapped to the variable.