# Catalog Server

Microservice to manage business catalogs.

**Note**: This Service uses a Json file named “config.json” as its configuration file.

## Technologies

* Back: NodeJS
* Database: MongoDB
* Messaging: Kafka

## Basic Response Model

* All responses are JSON objects
* All responses must have at least these two parameters
  + status: “success” or “failed”
  + message: String
* All responses must include response code (200 for success, non 200 for failures)

## Part 1: Authorization

Use Authorization Client library to provide access to Authorization server.

In this document, wherever mentioned “Authorization” refers to Authorization client library.

## Part 2: Catalog APIs

/catalog/create

Creates a new Catalog.

### Parameters

* user\_id: String
* bid: String
* dep\_id: String
* catalog\_info: JSON Object

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/create”, {user\_id:user\_id, bid:bid, dep\_id:dep\_id}) to check authorization
2. Set pid = IdGenerator.getNextId()
3. Set gid = CodeGenerator.getNextCode(unique\_name)
4. Store catalog in database
5. Publish event on Kafka
   1. Topic: “catalog\_created”
   2. Content:
      1. user\_id
      2. unique\_name
      3. bid
      4. pid
      5. gid
      6. dep\_id
      7. <catalog\_info fields>
      8. created\_at
      9. created\_by
6. Return Success

/catalog/edit

Edits a catalog.

### Parameters

* user\_id: String
* bid: String
* pid: String
* catalog\_info: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/edit”, {user\_id:user\_id, bid:bid, pid}) to check authorization
2. Update catalog using catalog\_info, bid and pid
3. Publish event on Kafka
   1. Topic: “catalog\_updated”
   2. Content:
      1. user\_id
      2. pid
      3. catalog\_info
      4. bid
4. Return Success or Failed

/catalog/delete

Deletes a catalog.

### Parameters

* user\_id: String
* bid: String
* pid: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/delete”, {user\_id:user\_id, bid:bid, pid:pid}) to check authorization
2. Delete catalog from catalogs table
3. Publish event on Kafka
   1. Topic: “catalog\_deleted”
   2. Content:
      1. user\_id
      2. pid
      3. bid
4. Return Success or Failed

/catalog/list

List all catalogs of businees

### Parameters

* user\_id: String
* bid: String

### Returns

* List of catalogs

### Steps

1. Call Authorization.authorize(“/catalog/list”, {user\_id:user\_id, bid:bid}) to check authorization
2. Get all catalogs with same bid
3. Return List

/catalog/listByCategory

List all catalogs of business based on category.

### Parameters

* user\_id: String
* bid: String
* category: String

### Returns

* List of catalogs

### Steps

1. Call Authorization.authorize(“/catalog/list/cat”, {user\_id:user\_id, bid:bid, category}) to check authorization
2. Get all catalogs with same bid and category
3. Return List

/catalog/listByDepartment

List all catalogs of business based on department.

### Parameters

* user\_id: String
* bid: String
* dep\_id: String

### Returns

* List of catalogs

### Steps

1. Call Authorization.authorize(“/catalog/list/dep”, {user\_id:user\_id, bid:bid, dep\_id:dep\_id}) to check authorization
2. Get all catalogs with same bid and dep\_id
3. Return List

/catalog/listByTags

List all catalogs of business based on tags.

### Parameters

* user\_id: String
* bid: String
* tags: Array of string

### Returns

* List of catalogs

### Steps

1. Call Authorization.authorize(“/catalog/list/tag”, {user\_id:user\_id, bid:bid, category}) to check authorization
2. Get all catalogs with same bid containing same tags
3. Return List

## Part 3: Set APIs

/set/create

Creates a new Catalog set.

### Parameters

* user\_id: String
* bid: String
* title: String
* desc: String
* filter: Json Object
* args: Json Object

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/set/create”, {user\_id:user\_id, bid:bid}) to check authorization
2. Set set\_id = IdGenerator.getNextId()
3. Store catalog set in catalog\_sets table
4. Publish event on Kafka
   1. Topic: “catalog\_set\_created”
   2. Content:
      1. set\_id
      2. title
      3. desc
      4. bid
      5. filters
      6. args
      7. created\_at
      8. created\_by
5. Return Success

/set/delete

Deletes a Catalog set.

### Parameters

* user\_id: String
* bid: String
* set\_id: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/set/delete”, {user\_id:user\_id, bid:bid, set\_id:set\_id}) to check authorization
2. Delete set from catalog\_sets table with same bid and set\_id
3. Publish event on Kafka
   1. Topic: “catalog\_set\_deleted”
   2. Content:
      1. set\_id
      2. bid
      3. created\_at
      4. created\_by
4. Return Success

/set/info

Return details of a set.

### Parameters

* user\_id: String
* bid: String
* set\_id: String

### Returns

* Catalog set info

### Steps

1. Call Authorization.authorize(“/catalog/set/info”, {user\_id:user\_id, bid:bid, set\_id:set\_id}) to check authorization
2. Get set from catalog\_sets table with same bid and set\_id
3. Return set info

/set/list

List all sets of this business.

### Parameters

* user\_id: String
* bid: String

### Returns

* List of sets

### Steps

1. Call Authorization.authorize(“/catalog/set/info”, {user\_id:user\_id, bid:bid}) to check authorization
2. Get all catalog sets from catalog\_sets table with same bid
3. Return list

## Part 3: Filter APIs

/filter/create

Creates a new Filter.

### Parameters

* user\_id: String
* title: String
* desc: String
* override: String
* rule: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/filter/create”, {user\_id:user\_id}) to check authorization
2. Set filter\_id = IdGenerator.getNextId()
3. Store filter in filters table
4. Publish event on Kafka
   1. Topic: “catalog\_filter\_created”
   2. Content:
      1. filter\_id
      2. title
      3. desc
      4. override
      5. rule
      6. created\_at
      7. created\_by
5. Return Success

/filter/delete

Deletes a Filter.

### Parameters

* user\_id: String
* filter\_id: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/catalog/filter/delete”, {user\_id:user\_id, filter\_id:filter\_id}) to check authorization
2. Delete filter from filters table with same filter\_id
3. Publish event on Kafka
   1. Topic: “catalog\_filter\_deleted”
   2. Content:
      1. filter\_id
      2. created\_at
      3. created\_by
4. Return Success

/filter/list

List all filters.

### Parameters

* user\_id: String

### Returns

* List of filters

### Steps

1. Call Authorization.authorize(“/catalog/filter/list”, {user\_id:user\_id }) to check authorization
2. Get all filters from filters table
3. Return list