# Sale Channels Server

Microservice to manage business sale channels.

**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: Channel APIs

/channel/create

Creates a new Catalog.

### Parameters

* user\_id: String
* bid: String
* title: String
* desc: String
* type: String
* sub\_type: String
* atrribs: String array

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/create”, {user\_id:user\_id, bid:bid}) to check authorization
2. Set chid = IdGenerator.getNextId()
3. Store channel in database
4. Publish event on Kafka
   1. Topic: “sale\_channel\_created”
   2. Content:
      1. user\_id
      2. bid
      3. chid
      4. title
      5. desc
      6. type
      7. sub\_type
      8. attribs
      9. created\_at
      10. created\_by
5. Return Success

/channel/edit

Edits a channel.

### Parameters

* user\_id: String
* bid: String
* chid: String
* channel\_info: String

### Returns

* <Basic Response>

### Steps

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

/channel/delete

Deletes a channel.

### Parameters

* user\_id: String
* bid: String
* chid: String

### Returns

* <Basic Response>

### Steps

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

## Part 3: Device APIs

/device/add

Add device to the channel.

### Parameters

* user\_id: String
* bid: String
* chid: String
* title: String
* desc: String
* purpose: String
* dev\_id: String
* dev\_serial: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/device/add”, {user\_id:user\_id, bid:bid, chid:chid, dev\_id, dev\_serial, purpose:purpose}) to check authorization
2. Store device details in channel\_devices table
3. Publish event on Kafka
   1. Topic: “sale\_channel\_device\_added”
   2. Content:
      1. user\_id
      2. bid
      3. chid
      4. title
      5. desc
      6. purpose
      7. dev\_id
      8. dev\_serial
      9. created\_at
4. Return Success

/device/remove

Remove device from channel.

### Parameters

* user\_id: String
* bid: String
* chid: String
* dev\_id: String
* dev\_serial: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/device/remove”, {user\_id:user\_id, bid:bid, chid:chid, dev\_id, dev\_serial }) to check authorization
2. Remove device details from channel\_devices table
3. Publish event on Kafka
   1. Topic: “sale\_channel\_device\_removed”
   2. Content:
      1. user\_id
      2. bid
      3. chid
      4. dev\_id
      5. dev\_serial
      6. created\_at
4. Return Success

/device/list

List devices of channel.

### Parameters

* user\_id: String
* bid: String
* chid: String

### Returns

* List of devices {dev\_id, dev\_serial, title, desc, purpose}

### Steps

1. Call Authorization.authorize(“sale\_channel/device/list”, {user\_id:user\_id, bid:bid, chid:chid}) to check authorization
2. Get all devices with same bid and chid from channel\_devices table
3. Return List of devices {dev\_id, dev\_serial, titel, desc, purpose}

/device/info

Get information about device in the channel

### Parameters

* user\_id: String
* bid: String
* dev\_id: String
* dev\_serail: String

### Returns

* Device info

### Steps

1. Call Authorization.authorize(“sale\_channel/device/info”, {user\_id:user\_id, bid:bid, dev\_id:dev\_id, dev\_serial:dev\_serial}) to check authorization
2. Get row from channel\_devices with same bid, dev\_id, dev\_serial
3. Return device info

## Part 4: Catalog Set APIs

/catalogset/assign

Select a catalog set for a channel.

### Parameters

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

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/sets/assign”, {user\_id:user\_id, bid:bid, chid:chid, set\_id:set\_id }) to check authorization
2. Store details in channel\_sets table
3. Publish event on Kafka
   1. Topic: “sale\_channel\_set\_assigned”
   2. Content:
      1. user\_id
      2. bid
      3. chid
      4. set\_id
      5. created\_at
4. Return Success

/catalogset/revoke

Remove set from channel.

### Parameters

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

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/sets/revoke”, {user\_id:user\_id, bid:bid, chid:chid, set\_id:set\_id }) to check authorization
2. Remove row in channel\_sets table with same bid, chid and set\_id
3. Publish event on Kafka
   1. Topic: “sale\_channel\_set\_revoked”
   2. Content:
      1. user\_id
      2. bid
      3. chid
      4. set\_id
      5. created\_at
4. Return Success

/catalogset/info

Get set info of a channel.

### Parameters

* user\_id: String
* bid: String
* chid: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“sale\_channel/sets/info”, {user\_id:user\_id, bid:bid, chid:chid }) to check authorization
2. Get a set from channel\_sets table using bid and chid
3. Return Success