# Ecom Accounts Server

Microservice to manage ecommerce accounts linked to this business.

**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: Cipher

Use Cipher Engine Client library to provide access to Cipher Engine Server.

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

**Note**: In your code create an empty class with same name and functions for Cipher library. Later we will replace it with the real Cipher lib.

## Part 3: Ecommerce accounts APIs

/ecomaccount/add

Add a new ecommerce account to business.

### Parameters

* user\_id: String
* bid: String
* type: String
* subtype: String
* title: String
* desc: String
* username: String
* password: String
* payload: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/ecomaccounts/add”, {user\_id, bid, type, subtype}) to check authorization
2. Create credential\_json using parameters
   1. username: username
   2. password: password
   3. payload: payload
3. Create credential\_string by converting credential\_json to String
4. Set request\_id = IdGenerator.getNextId()
5. Call ciphered = Cipher.encrypt(credential\_string, request\_id, “AES256”)
6. Store account in database
   1. account\_id: New Random UUID
   2. type: type
   3. subtype: subtype
   4. title: title
   5. desc: desc
   6. credentials: ciphered.content
   7. lockstore\_ref: ciphered.reference
   8. request\_id: request\_id
   9. bid: bid
7. Publish event on Kafka
   1. Topic: “social\_account\_added”
   2. Content:
      1. user\_id
      2. bid
      3. type
      4. subtype
      5. title
      6. desc
      7. created\_at
8. Return Success

/ecomaccount/edit

Edits an account.

### Parameters

* user\_id: String
* bid: String
* account\_id: String
* title: String
* desc: String
* username: String
* password: String
* payload: String

### Returns

* <Basic Response>

### Steps

1. Call Authorization.authorize(“/ecomaccounts/edit”, {user\_id, bid, account\_id}) to check authorization
2. Find an account with same account\_id and edit {title, desc, credentials} fields.
   1. You need to call Cipher again to create credentials string.
   2. Please refer to /account/add
3. Return Success or Failed.

/ecomaccount/delete

Deletes an account.

### Parameters

* user\_id: String
* bid: String
* account\_id: String

### Returns

* <Basic Response>

### Steps

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

/ecomaccount/list

List all accounts of business

### Parameters

* user\_id: String
* bid: String

### Returns

* List of accounts

### Steps

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

/ecomaccount/get

Get account with same ID

### Parameters

* user\_id: String
* bid: String
* account\_id: String

### Returns

* Account

### Steps

1. Call Authorization.authorize(“/ecomaccounts/get:byId”, {user\_id, bid, account\_id) to check authorization
2. Get account with same account\_id and bid
3. Call Cipher.decrypt(account.credentials, account.request\_id, account.lockstore\_ref) to get deciphered\_credentials json String
4. Return JSON Object with these fields
   1. account\_id: account.account\_id
   2. type: account.type
   3. subtype: account.subtype
   4. title: account.title
   5. desc: account.desc
   6. credentials: deciphered\_credentials
   7. bid: bid

/ecomaccount/getByType

Get account with same type.

### Parameters

* user\_id: String
* bid: String
* type: String
* subtype: String

### Returns

* Account

### Steps

1. Call Authorization.authorize(“/ecomaccounts/get:byType”, {user\_id, bid, type, subtype) to check authorization
2. Get account with same type, subtype and bid
3. Call Cipher.decrypt(account.credentials, account.request\_id, account.lockstore\_ref) to get deciphered\_credentials json String
4. Return JSON Object with these fields
   1. account\_id: account.account\_id
   2. type: account.type
   3. subtype: account.subtype
   4. title: account.title
   5. desc: account.desc
   6. credentials: deciphered\_credentials
   7. bid: bid