CleverTap SDK Documentation

This project's source(*src*) is divided into 8 sub packages:

- http
- helper
- profile
- event
- campaign
- report
- payload
- response

1. http:

This contains the singleton Apache http client class for handling the API requests.
getRequest() and *postRequest()* methods defined in *HttpClient* class are used to make Get and Post requests respectively to API endpoints.
In both methods we add request configuration using Apache *RequestConfig*. Using this we set the timeouts for the connection.
CloseableHttpClient is used for making requests, to which we have added connection pooling manager and request retry handling. In case during the request, if some exception occurs or if we receive server code above 429 or 500 or above then the request is retried after a fixed interval. Currently this is set for maximum 3 retries with a gap of 3 seconds. Currently in connection manager only 3 connections per route are set.
With the connection manager we have also set *idleConnectionMonitorThread* to identify any passive connections and remove them to free up the resources.
All these are set when the constructor of *HttpClient* is called.
The Clevertap account id and password are required in request header which are set using static reference from clevertap instance class in Helper package.

Important source:
https://hc.apache.org/httpcomponents-client-4.5.x/current/tutorial/html/fundamentals.html

2. helper:

This package contains the *ClevertapInstance*, *ObjectMapperHelper* classes and a subpackage of enums used in the project.
ClevertapInstance defines methods to instantiate the *Profiles*, *Events*, *Reports* and *Campaigns* classes. Clevertap account ID and password are defined to be static variables here. Region for our API endpoints is also defined here using *Region* enum from the Enum sub package.
ObjectMapperHelper is a class which defines our object mapper which is used to create the json object from payload classes as well as response class objects from json response. This class will be called in the *Profiles*, *Events*, *Reports* and *Campaigns* classes for mapping of objects.
Enums sub package also contains *CampaignMethod* and *UserIdentity* enums which are used in *createCampaignTargetId* method in *Campaigns* class and *getUserProfileById* method of *Campaigns* class respectively.

3. profile:

This package contains *Profiles* class which has methods for making api calls related to profiles. This is instantiated in the *ClevertapInstance* class.

Methods:

1. *uploadUserProfile*: This is used for uploading user profiles. A list of profile data is given as an argument and in response it gives an object of type *Response*.
2. *getUserProfileCursor*: This method is used as the first step to retrieve the profiles data based on events information. We provide the filters related to what profiles data we want to retrieve along with the batch size and other flags like app, events, etc., for including or excluding these data in response. It's response is a *Cursor*(String) which is required for making a second call to api endpoint to get the profile data.
3. *getUserProfileData*: This method is the second step to retrieve the user profile data. We pass the cursor we got from the above method to this method and in response it provides us the batch of profile data as an object of *GetUserProfileResponse*.
4. *getUserProfileById*: This method is used to retrieve user profile data based on specific user ids. We pass the type and value of user id to this method and in response get an object of *GetUserProfileResponse*.

5. *uploadDeviceTokens*: This method is used to add an existing device token to a CleverTap user profile. List of token data is passed as an argument to this method.
6. *getProfileCount*: This method is used to get the total number of profiles that match an event query.
7. *getProfileCountByReqId*: Sometimes the *getProfileCount* may receive a partial response from the server and along with it a request id is also received. We use this request id as an argument for the *getProfileCountByReqId* method to receive the count response.+
8. *deleteUserProfile*: This method is used to delete a specific user profile, which is passed as an argument to this method.
9. *demergeUserProfile*: This method enables us to demerge a user profile. We pass identities as arguments to this method.
10. *subscribe*: The method provides the ability to set a phone number or email status as subscribed or unsubscribed. We pass type, value and category in an object list as an argument to this method.
11. *disassociate*: This method enables us to disconnect a phone number from a user profile. We pass type and value in object list to this method.

4. event:

This package contains *Event* class which has methods for making api calls related to events. This is instantiated in the *ClevertapInstance* class.

Methods:

1. *uploadEvents*: This is used for uploading user events. A list of events data is given as an argument and in response it gives an object of type *Response*.
2. *getEventsCursor*: This method is used as the first step to retrieve the events data based on events information. We provide the filters related to what event data we want to retrieve along with the batch size and other flags like app, profiles, etc., for including or excluding these data in response. It's response is a *Cursor*(String) which is required for making a second call to api endpoint to get the events data.
3. *getEventsData*: This method is the second step to retrieve the user events data. We pass the cursor we got from the above method to this method and in response it provides us the batch of events data as an object of *GetEventsResponse*.

4. *getEventCount*: This method is used to get the total number of events in a specified duration.
5. *getEventCountByReqId*: Sometimes the *getEventCount* may receive a partial response from the server and along with it a request id is also received. We use this request id as an argument for the *getEventCountByReqId* method to receive the count response.

5. campaign:

This package contains *Campaigns* class which has methods for making api calls related to campaigns. This is instantiated in the *ClevertapInstance* class.

Methods:

1. *createCampaignTargetUserEvents*: This method is used to create a campaign, we have to specify the message, channel, and people to target by their user events and properties in the payload argument. It receives a success or failure response back from the server.
2. *createCampaignTargetId*: This method is used to send notifications to users based on their Facebook ID, Email ID, custom-defined identity, or CleverTap ID. We need to provide a campaign method(SMS, email, etc.) as an argument along with the campaign data.
3. *stopScheduledCampaign*: This method lets us stop a scheduled/running campaign. We need to provide campaign id as an argument to it.
4. *getCampaignReport*: This method lets us get campaign metrics. We specify the id of the report needed in arguments. If the campaign has not completed yet, we will receive an error message indicating that status.
5. *getCampaigns*: This method is used to get the campaigns created in a specific date range. We provide a date range as an argument to this method.

6. report:

This package contains *Report* class which has methods for making api calls related to reports. This is instantiated in the *ClevertapInstance* class.

Methods:

1. *getRealTimeCounts*: This method is used to get a real-time count of active users in the past five minutes in our app. An empty payload argument will return a real-time count of active users. Otherwise If user_type is set to true, then it will return the split of users by type.
2. *getMessageReports*: This method is used to download a list of messages sent by CleverTap. We need to pass the date range to the method. We can also filter these results by setting optional parameters that will return only specific channels or message statuses.
3. *getTopPropertyCount*: This method is used to retrieve counts for the most and least frequently occurring properties for a particular event in a specified duration. We need to pass event name, date range and groups to the method to get the count.
4. *getTopPropertyCountByReqId*: Sometimes the *getTopPropertyCount* may receive a partial response from the server and along with it a request id is also received. We use this request id as an argument for the *getTopPropertyCountByReqId* method to receive the count response.
5. *getTrends*: This method is used to retrieve daily, weekly, and monthly event trends in a specified duration. We need to pass event name, date range and groups to the method to get the count.

7. payload:

This package contains POJO classes used to create payload objects in methods described in above packages. Internally, these POJO classes are converted to json objects using object mappers.

- *Cursor*: It's object is used to store the cursor string received from server when methods *getEventsCursor* in *Event* class and *getUserProfileCursor* in Profiles class are used.
- *EventPropertyFilter*: It's object is used in the *EventPayload* and *ProfilePayload* instance to define event properties based on which we will get filtered results from the server in the *getEventCount* method of *Event* class and *getProfileCount* method of *Profiles* class.

payload subpackages:

profile: All POJO classes of this sub package are used to create payloads for methods of *Profiles* class.

- *ProfilePayload*: It's object is used in methods of *Profiles* class for creating payload.
- *TokenData*: It's object is used in *ProfilePayload* instance to upload device token in method *uploadDeviceTokens*.
- *Keys*: It's object is used in the TokenData instance to create keys for token data for chrome.
- **profiledata(child package of profile subpackage)**:
    + *ProfileData*: It's object is used in *ProfilePayload* instance for method *uploadUserProfile* in *Profiles* class to add user profile data in payload.
    + *CategoryResubscribe*: It's object is used in *ProfileData* instance to set users to resubscribe to categories.
    + *CategoryUnsubscribe*: It's object is used in *ProfileData* instance to set users to unsubscribe to categories.

event: All POJO classes of this sub package are used to create payloads for methods of *Event* class.

- *EventPayload*: It's object is used in methods of *Event* class for creating payload.

report: All POJO classes of this sub package are used to create payloads for methods of *Report* class.

- *ReportPayload*: It's object is used in methods of *Report* class for creating payload.
- *Group*: It's object is used in *ReportPayload* instance for creating groups hashmap objects where key is the user defined group name and value is *Group* object. This is used in *getTopPropertyCount* and *getTrends* methods of *Report* class.

campaign: All POJO classes of this sub package are used to create payloads for methods of *Campaigns* class.

- *CampaignPayload*: It's object is used in methods of *Campaigns* class for creating payload.
- *Where*: It's object is used in the *CampaignPayload* instance to filter users for campaigns. It is used in the *createCampaignTargetUserEvents* method of *Campaigns* class.
- *CommonProfileProperties*: It's object is used in *Where* instance to set profile property for filtering users.
- *ProfileFields*: It's object is used in the *CommonProfileProperties* instance for setting properties of profiles for filters.

- *To*: It's object is used in the *CampaignPayload* instance to create campaigns based on user ids. It is used in the *createCampaignTargetId* method of *Campaigns* class.
- *ControlGroup*: It's object is used in the *CampaignPayload* instance to create the campaigns based on user events if the custom control group is enabled from the dashboard. It is used in the *createCampaignTargetUserEvents* method of *Campaigns* class.
- *DateRangePayload*: It's object is used as a payload to *getCampaigns* method of *Campaigns* class. It specifies the date range to get the campaigns from.

- **campaigncontent(child package of campaign subpackage)**:
    + *Content*: It's object is used in the *CampaignPayload* instance to define the content of the message. This is used in *createCampaignTargetUserEvents* method and *createCampaignTargetId* method of *Campaigns* class.
    + *CampaignContentTitle*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object instead of a string for app inbox campaigns.
    + *PlatformSpecificContent*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns.
    + *CampaignContentMessage*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object instead of a string for app inbox campaigns.
    + *CampaignContentMedia*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns.
    + *CampaignContentIcon*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns for messages with icons.
    + *CampaignContentAttachments*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for whatsapp campaigns with attachments.

    - **actions(child package of campaigncontent subpackage)**:

        + *Action*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns.

+ *ActionLinks*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns.

- **actionurl(child package of actions subpackage)**:
  + *ActionUrl*: It's object is used in the *Content* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns.
  + *AndroidActionUrl*: It's object is used in the *ActionUrl* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns for action urls for android.
  + *IosActionUrl*: It's object is used in the *ActionUrl* instance for *createCampaignTargetUserEvents* method of *Campaigns* class. This is an object for app inbox campaigns for action urls for android.

- **plateformtype(child package of campaigncontent subpackage)**:
  + *PlatformTypeAndroid*: It's object is used in the *PlatformSpecificContent* instance for *createCampaignTargetUserEvents* method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns for android.
  + *PlatformTypeIos*: It's object is used in the *PlatformSpecificContent* instance for *createCampaignTargetUserEvents* method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns for iOS devices.
  + *PlatformTypeChrome*: It's object is used in the *PlatformSpecificContent* instance for *createCampaignTargetUserEvents* method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns for chrome.
  + *PlatformTypeFirefox*: It's object is used in the *PlatformSpecificContent* instance for *createCampaignTargetUserEvents* method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns for firefox.
  + *PlatformTypeSafari*: It's object is used in the *PlatformSpecificContent* instance for

createCampaignTargetUserEvents method and the *createCampaignTargetId* method of *Campaigns* class. This is used for push notification(web and mobile) campaigns for safari.

8. response:

This package contains POJO classes used to receive response objects in methods described in above packages. Internally, json response is converted to these POJO class objects.

Response: This is the main class to handle all the major responses from the server. This class is extended in other classes as some of its methods are used in its subclasses. Below classes are used as objects in *Response* class:

- *Target*: It's object is used to store data received in the *getCampaigns* method in campaigns class.
- *Message*: It's object is used to store data received in the getMessageReports method in Report class.
- *Result*: It's object is used to store data received in the *getCampaignReport* method in campaigns class.
- *OtherInfo(HashMap)*: In case there is some data in response from the server which is not defined in our POJO class, the object mapper will put that data in this hashmap.

GetUserProfileResponse: This class is used in handling the response in retrieving user profile data in *getUserProfileData* method of profiles class. *Response* is extended into this class. Below class is used as object in this class:

- *RecordsProfile*: This class is used as an object in *GetUserProfileResponse* to store the records object received in profiles data response.

GetEventsResponse: This class is used in handling the response in retrieving events data in the *getEventsData* method of events class. *Response* is extended into this class. Below class is used as object in this class:

- *RecordsEvent*: This class is used as an object in *GetEventsResponse* to store the records object received in events data response. Below class is used as object in this class:
    - *Profile*: It's object is used in *RecordsEvent* to store the profile data.

Note:

We also have a *test* package along with the *src* package. The *test* package contains unit and integration tests of *src* classes and it's methods. We have used junit 5 for testing.