HCLSoftware

HCL CDP v12.1.10 Administrator Guide

Contents

Chapter 1. Introduction..... 3
Chapter 2. Onboard Tenant..... 4
Create Tenant Account..... 4
Onboard Tenant Account..... 6
Create User Profile..... 9
Chapter 3. Enable Channel Modules..... 11
Chapter 4. SST Enhancement 12

Chapter 1. Introduction

This section provides an overview of how to create, on-board, and manage tenant accounts effectively. You will also learn how to enable channels and modules for each tenant, ensuring they have the necessary tools and functionalities to maximize their experience within the CDP.

Chapter 2. Onboard Tenant

Onboarding a tenant is the first step in configuring a new client environment within the HCL CDP. It involves creating a dedicated tenant account and provisioning initial administrative users who will manage the platform. This process enables data, access controls, and configurations to be logically separated for each organization, ensuring secure and scalable operations.

As part of tenant onboarding, administrators perform two critical tasks:

- Creating a tenant account using the admin portal, which establishes the core environment.
- Adding and managing users who will access and operate the system under that tenant.

These tasks must be completed in sequence to ensure that the tenant environment is fully functional and accessible..

Create Tenant Account

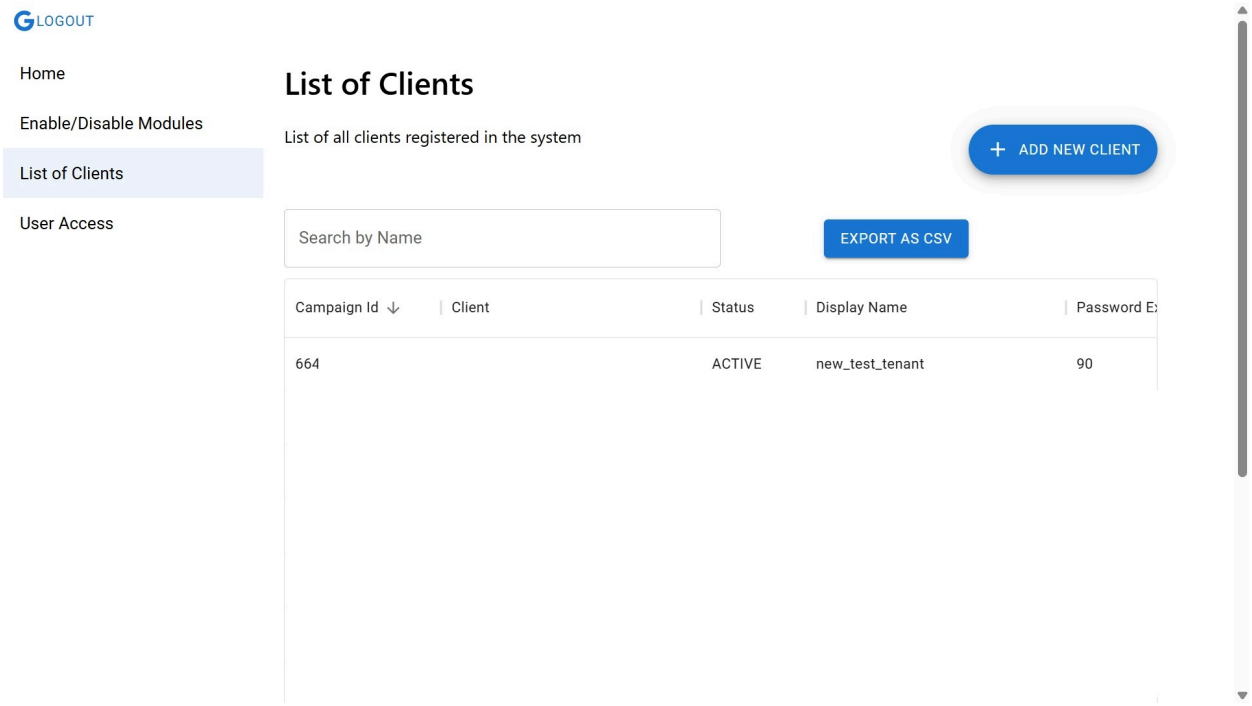
After successfully installing the application, the next crucial step is setting up tenant accounts. A tenant account acts as a dedicated environment within the system, allowing organizations or user groups to operate independently and securely.

Creating tenant accounts is typically handled by admin users. In cloud environments like AWS, this task is performed by the cloud admin team. In OpenShift environments, customer-side administrators are responsible for tenant creation. Using the admin portal, these authorized users can define tenant-specific settings, assign administrative roles, and configure access controls, ensuring each tenant operates within a secure and isolated environment tailored to their needs.

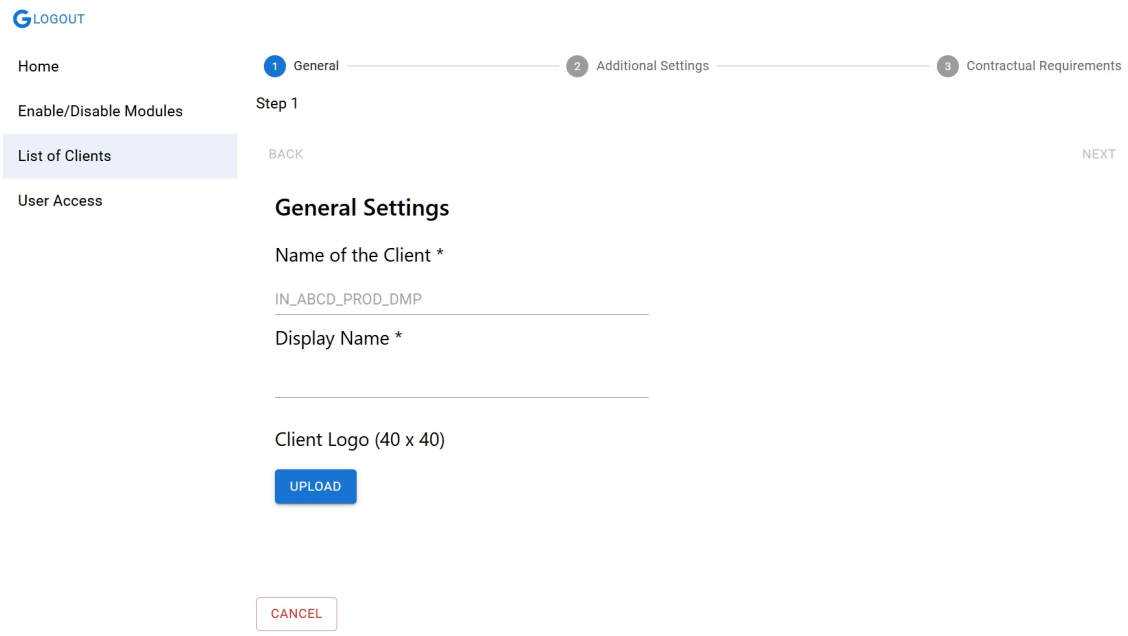
Create a Tenant account

To create a tenant account in the HCL CDP Admin portal, follow the steps below:

1. Log in to the Admin portal using the Admin credentials, and click **List of Clients**.



2. In the **List of Clients** page, click **ADD NEW CLIENT**.



3. In the **General** section, enter the name of the client and display name. Then, click **Upload** to upload the client's logo, in 40x40 size.
4. Click **Next**, and in the **Additional Settings** section, provide domain details.

5. In the **Set Password Expiry in days** field. enter the number of days for setting the number of days before a password reset is required.
6. Similarly, click **Next** and enter Contractual Requirements like number of user profiles, contract ending date, and so on.

Onboard Tenant Account

After creating a tenant account in the admin portal, backend onboarding is required to complete the setup process. This step ensures that the tenant is fully recognized by the system and properly integrated across all supporting services.

To enable full functionality, the tenant account must be registered in the application database. For clients with an associated campaign ID, this identifier must be inserted into the database and propagated across all dependent services. These tasks are typically performed by platform administrators or backend operations teams to ensure smooth and secure onboarding.

Onboard Tenant Account in DB

To onboard the campaign ID (tenant account), follow the steps below:

1. In the database, update the following queries in sequence to onboard the tenant account. As a first step, select a vrm database.

   ```
   Use vrm;
   ```

2. Set the Tenant ID /Campaign ID to the campaign variable. Make sure this tenant is created in the Admin UI.

   ```
   set @campaign := <campaignId>;
   ```

3. Set the vertical name, which can be banking or insurance. This name is selected in the Admin UI while creating the Tenant account.

   ```
   set @dmpVerticalId := (select Id from DMPCampaignVertical where Name='<VerticalName>');
   UPDATE `Campaign` SET IsDMP=1, IsRealtimeSeg=1, IsBfsi=1 WHERE Id=@campaign;
   ```

4. Set the AWS region of the tenant account in the region variable.

   ```
   set @region:= "aps1";
   ```

5. Update the AWS region in the DMPCampaign table.

   ```
   Update DMPCampaign set Region = 'aps1' where CampaignId = @campaign;
   ```

6. To support Unicode characters, @campaign is tenantid.

   ```
   insert into `CampaignUnicodeRangeMapping` (CampaignId, UnicodeRanges) values (@campaign,
    'U+0020-U+007F');
   ```

7. After updating the vertical name, generate a KMS key using the AWS console. Then, attach a key policy that allows the EC2 and Kubernetes service account IAM role to access the created KMS Key ID. Similarly, insert KMS key Id for Hashicorp(Openshift into CampaignEncryptionKeyMapping. This key ID will be used for PII (e.g. Mobile/Phone) Data Encryption.

   ```
   insert into `CampaignEncryptionKeyMapping` (CampaignId, keyId, TTL_Seconds, Keys_Count, IsActive) values
    (@campaign, '<KMS KEY ARN>', 86400, 100, 1);
   ```

8. If needed, expose the segmentation parameters on UI as mentioned below. To enable the segmentation in fields like Country customer properties/attributes, add entries in FieldAliasTable table for attribute in profile, where adv_id is

tenantid, original_field_name is data point code, modified_field_name is UI field name, type_of_ui_widget is data type of field like Text/Number, json_key is json path.

```
INSERT Into FieldAliasTable
 ( adv_id,original_field_name,modified_field_name,type_of_ui_widget,is_new_user,json_key,index_of_group_
key,suggest_at,CreatedOn,UpdatedOn,CreatedBy,UpdatedBy)
 VALUES ('@campaign','COUNTRY','COUNTRY','Text',0,'country',0,0,'2024-04-12',NULL,<user ID>,NULL),
 ('@campaign','CITY','CITY','Text',0,'city',0,0,'2024-04-20',NULL,<user ID>,NULL);
```

Create Index in MongoDB

After onboarding the tenant account in db, create indexes on the collections used for segmentation and suggestion/count caching in the MongoDB. To create indexes in MongoDB, follow the steps below:

1. Log in to the primary node using mongosh, and execute the below queries in sequence. Switch the segmentation database.

   ```
   use segmentation;
   ```

2. Create a hashed index for segmentation purposes.

   ```
   db.VIZVRM<CampaignId>.createIndex({"key":"hashed"},{"sparse":true});
   ```

3. Now, switch cache database, and create an index on the queryId field for storing count or metrics data for that campaign.

   ```
   use cache;
   db.VIZVRM<CampaignId>_count.createIndex({"queryId":1},{"sparse":true,"unique":true});
   ```

4. Create an index on the queryId field for storing suggestion related data.

   ```
   db.VIZVRM<CampaignId>_suggestion.createIndex({"queryId":1},{"sparse":true,"unique":true});
   ```

Add Campaign ID in Devtron

After onboarding the campaign ID into the database, the next step is to update the backend application services to ensure proper integration. Specifically, the campaign ID must be added to configuration properties in the `dmp-sst` and `core-api` applications.

To Add campaign ID in the Devtron, follow the steps below:

1. Log into the Devtron app, in the configuration of dmp-sst app, set the campaign ID in the CampaignIncludeList Property, and deploy the app.
2. Similarly, in the configuration of core-api app, set the campaign ID in the CampaignRegionList Property, and deploy the app.

Data Source Configurations

Configure the MySQL data sources with campaign details like campaign ID, email, mobile and so on.

To update the Mysql DB with the data source configurations, follow the steps below:

1. Set the User Identifier for Upset charts, and insert the campaign details in the CDPCampaignUserIdentifier table using the following queries in sequence.

```
insert into CDPCampaignUserIdentifier (CampaignId,Code,AlternateDescription) values
 (<CampaignId>,'vizid','Cookie');
insert into CDPCampaignUserIdentifier (CampaignId,Code,AlternateDescription) values
 (<CampaignId>,'crmid','Customer_Id');
insert into CDPCampaignUserIdentifier (CampaignId,Code,AlternateDescription) values
 (<CampaignId>,'he','Email');
insert into CDPCampaignUserIdentifier (CampaignId,Code,AlternateDescription) values
 (<CampaignId>,'hm','Mobile');
commit;
```

2. Update the DMPCampaignDataDictionary table with Campaign ID details to set the primary identifier for the tenant:

```
Update DMPCampaignDataDictionary set Preference='pri' where Id = <crmAttributeId> and CampaignId =
 <CampaignId>;
```

3. Set the secondary identifier and priority of identifier for the tenant using the below queries. The <crmAttributeId>,<emailAttributeId> and <phoneAttributeId> are from DMPCampaignDataDictionary table.

```
Update DMPCampaignDataDictionary set Preference='sec' where Id = <emailAttributeId> and CampaignId =
 <CampaignId>;
Update DMPCampaignDataDictionary set Preference='sec',Priority=2 where Id = <phoneAttributeId> and
 CampaignId = <CampaignId>;
commit;
update DMPCampaignDataDictionary set IsRtsWhitelisted = 1 where Id = <crmAttributeId>;
```

4. Create a Data Ingestion (DI) API data source in the MySQL database using the following queries in sequence.

```
set @pss :=(select max(Id) from DMPCampaignDataSource)+1;
INSERT INTO DMPCampaignDataSource (Id, CampaignId, Name, Format, IsActive, IsNba, IsTriggerDataSource,
 DisplayName) VALUES
(@pss,<CampaignID>,'dataingestionapi','json',1, 1, 1, 'DIAPI');
```

5. Similarly, create a pixel listener data source using the following queries in sequence.

```
INSERT INTO DMPCampaignDataSource(CampaignId, Name, Format, IsActive, IsOnline, IsTriggerDataSource,
 DisplayName)
VALUES(<CampaignID>, 'analyze_post', 'json', 1, 1, 1, 'analyze_post');
```

Enable DI API Email and Phone Encryption

Enable Email and Phone Encryption in DI API:

1. Go to the DI API handler directory, and navigate to: `/MarketingAutomation/tree/release/DataIngestionAPIV2/js-server/handler`.
2. Switch to the correct branch to ensure you're on the `release` branch.
3. Locate and open the file named `<tenantid>.js`.
4. Add or update the logic to enable email and phone encryption according to your project's security requirements.
5. Commit the updated file to the `release` branch with an appropriate commit message, and deploy the updated app so the changes take effect.

Create Pixel Listener Data Source

To create a Pixel Listener Data Source, follow the steps below:

1. Connect to the database that hosts the `DMPCampaignDataSource` table.
2. Run the following SQL command, replacing `<CampaignID>` with the actual tenant ID to insert the pixel listener data source:

```
INSERT INTO DMPCampaignDataSource
(CampaignId, Name, Format, IsActive, IsOnline, IsTriggerDataSource, DisplayName)
VALUES(<CampaignID>, 'analyze_post', 'json', 1, 1, 1, 'analyze_post');
```

Refresh UI dashboard

After updating the data source, use the below url on any browser to refresh the UI dashboard,

```
https://<Domain>/populateCDPDashboardData/campaign/<CampaignId>
```

Sample url is provided below for reference.

```
https://<rts-ms domain>/populateCDPDashboardData/campaign/6525
```

Create User Profile

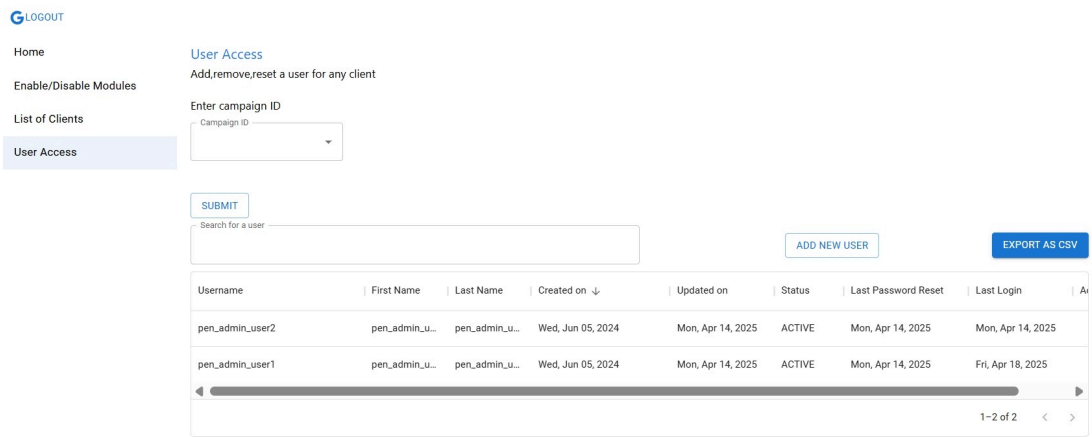
The User Access page in the admin portal provides administrators with essential tools to manage user accounts within a tenant. This functionality helps ensure proper access control and account lifecycle management.

From the User Access page, administrators can perform key actions such as:

- Create new user profiles to grant access to the platform.
- Activate or deactivate users based on operational needs or security requirements.
- Reset user passwords to maintain account security.
- Send email invitations to onboard new users quickly and efficiently.

To create a user profile, follow the steps below:

1. In the Admin portal, on the left pane, click **User Access**, and select a campaign ID.
2. Click **Submit**. As a result the list of available user profile will be displayed as shown below.



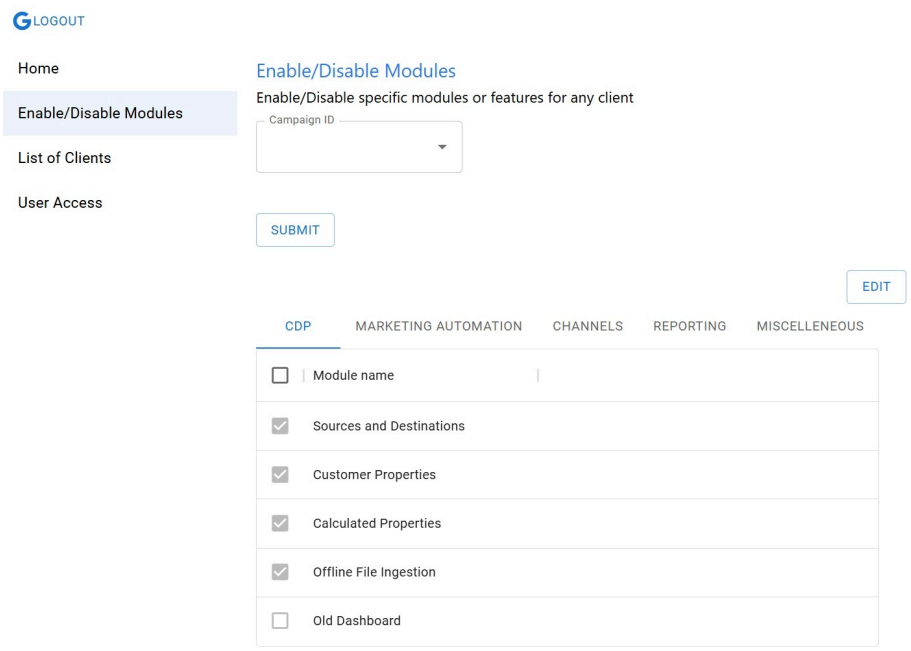
3. Click **ADD NEW USER** to add a new user profile.
4. In the **Add new user** page, enter user name and email address to send the user credentials to the user.
5. Click **SEND INVITE** to create the user profile to access the application. As a result, the user credentials will be emailed to the user email address. Also, the user profile will be listed in the user profile list.
6. Also, you can view the user login activities like last login, last password reset, and user details updated on.
7. At the end of the user profile, click the three horizontal icon, and you can perform password reset, edit profile or active/de-active the user profile.
8. Click **EXPORT AS CSV** to download all the user details in .csv file.

Chapter 3. Enable Channel Modules

After onboarding the tenant account, enable or disable modules required for the tenant based on the proposed contract.

To enable the modules, follow the steps below:

1. In the Admin portal, on the left pane, click **Enable/Disable Modules**.
2. Select a campaign ID that was created for the Tenant account in the previous step, and click **Submit**.
3. As a result, the list of modules grouped in the **CDP**, **MARKETING AUTOMATION**, **CHANNELS**, **REPORTING** and **MISCELLANEOUS** tabs are displayed as shown below.



4. From the list of modules, select the required modules, and click **SAVE**.

Chapter 4. SST Enhancement

To support enhanced user profile enrichment across the platform, both Pixel Listener and DI API data sources must be configured to accept and map key identity and demographic attributes. This section ensures consistent and accurate user data updates within the profile system for each campaign.

This section involves inserting mapping configurations into the `DMPCampaignDataSourceMapping` table. These mappings define how specific attributes from incoming JSON payloads—such as `customer_id`, `email`, `phone`, `name`, and location details—are extracted and used to update user profiles.

- For **Pixel Listener data sources**, attributes are typically extracted from the `properties` or `otherIds` JSON objects. Mappings include:
  - `customer_id` (from multiple possible keys)
  - `email`, `phone` (from `otherIds`)
  - `name` (from `properties`)
- For **DI API data sources**, mappings are set for attributes like:
  - `FIRST_NAME`, `LAST_NAME`, `email`, `mobile` (encrypted or hashed fields)
  - `CITY`, `STATE`, `COUNTRY`, and `customerId`

Each mapping entry requires the associated **DataSource ID** from the `DMPCampaignDataSource` table and the corresponding **Attribute ID** from the `DMPCampaignDataDictionary` table. Once mappings are defined, changes are committed to the database, allowing the ingestion pipeline to correctly identify and update user profiles using incoming event data.

Configure attribute mapping for Pixel Listener Data Source

To configure attribute mapping for Pixel Listener Data Source, follow the steps below:

1. Open your SQL client, and run the following query to accept `customer_id` and map to profile, replacing `<DataSourceId>` and `<AttributeId>` with actual values:

```
INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)

VALUES (<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__customer_id",
 "alternate_keys":["properties__userId","otherIds__customerId", "otherIds__customer_id"]}');
commit;
```

2. Accept and map `email` and `phone` from `otherIds`, and run the following insert statements:

```
INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES (<DataSourceID>, <AttributeId>, 0, 'UPDATE', '{"input_col": "otherIds__email"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES (<DataSourceID>, <AttributeId>, 0, 'UPDATE', '{"input_col": "otherIds__phone"}');
```

3. Accept and map `name` from `properties`, and run this insert statement:

```
INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES (<DataSourceID>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__name"}');
commit;
```

Configure attribute mapping for DI API Data Source

To configure attribute mapping for Pixel Listener Data Source, follow the steps below:

1. Map identity and profile attributes from `properties` JSON, and replace `<DataSourceId>` and `<AttributeId>` in each of the following statements:.

```
INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__FIRST_NAME"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__LAST_NAME"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "hashed_email"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "hashed_mobile"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__CITY"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__STATE"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__COUNTRY"}');

INSERT INTO DMPCampaignDataSourceMapping (DMPCampaignDataSourceId, DMPCampaignDataDictionaryId,
 IsMandatory, ProfileUpdateFunction, Metadata)
VALUES ((<DataSourceId>, <AttributeId>, 0, 'UPDATE', '{"input_col": "properties__customerId"}');

commit;
```