HCLSoftware

HCL CDP v12.1.8 Administrator Guide

Contents

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. Tenant Creation

After the application installation is complete, the next step is to create a tenant account. This account will enable you to access and manage the admin portal effectively.

Create a Tenant account

To create a tenant account in 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. Similarly, click next and enter Contractual Requirements like number of user profiles, contract ending date, and so on.

Chapter 3. Onboarding Tenant Account

After creating the tenant account in the Admin portal, on board the campaign ID of the client into database and update campaign ID changes across the services.

Onboard Tenant Account in DB

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

1. In the database, update the following queries in sequence to on board the tenant account.

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

2. Set the vertical name, which can be banking or insurance.

```
set @dmpVerticalId := (select Id from DMPCampaignVertical where Name='<VerticalName>');
set @region:= "aps1";
UPDATE `Campaign` SET IsDMP=1, IsRealtimeSeg=1, IsBfsi=1 WHERE Id=@campaign;
#INSERT INTO `DMPCampaign` (CampaignId, DMPCampaignVerticalId,Region, UseVertica) VALUES (@campaign,
 @dmpVerticalId,@region, 0);
Update DMPCampaign set Region = 'aps1' where CampaignId = @campaign;
insert into `CampaignUnicodeRangeMapping` (CampaignId, UnicodeRanges) values (@campaign,
 'U+0020-U+007F');
```

3. 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.

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

4. If needed, expose the segmentation parameters on UI as mentioned below.

```
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

Create indexes on the collections used for segmentation and suggestion/count caching in 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.

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

Add Campaign ID in Devtron

Update the dmp-sst and core-api apps with the Campaign ID in the Devtron.

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

1. 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 using the following queries in sequence:

```
Update DMPCampaignDataDictionary set Preference='pri' where Id = <crmAttributeId> and CampaignId =
 <CampaignId>;
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;
#<crmAttributeId>,<emailAttributeId> and <phoneAttributeId> are from DMPCampaignDataDictionary table.
update DMPCampaignDataDictionary set IsRtsWhitelisted = 1 where Id = <crmAttributeId>;
```

3. Create a DI API data source in 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');
```

4. 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');
```

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
```

Chapter 4. User Creation

In the admin portal, the user access page allows administrators to perform essential actions related to user accounts. You can create new user profiles, manage their activation and deactivation status, reset passwords, and send email invitations.

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.
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. At the end of the user profile, click the three horizontal icon, and you can perform password reset, edit profile or active/deactive the user profile.

Chapter 5. Channel Integration and Enablement

After creating the tenant account, enable or disable modules that are 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**, and select a campaign ID that was created for Tenant account in the previous step.
2. Click **Submit**. As a result, the list of modules grouped in the **CDP**, **MARKETING AUTOMATION**, **CHANNELS**, **REPORTING** and **MISCELLANEOUS** tabs is displayed.
3. From the list of modules, select the required modules, and click **SAVE**.

Chapter 6. SST Enhancement

Enhance the mapping of campaign data sources within the Pixel Listener and Data Integration (DI) framework.

To enhance the SST in Pixel Listener, follow the steps below:

1. Insert the campaign data source details in the DMPCampaignDataSourceMapping table. Use the following queries in sequence.

```
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;

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"}');

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

2. Similarly, update enhancements for DI framework using the following queries in sequence.

```
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;

#<DataSourceId> is from DMPCampaignDataSource Table and <AttributeId> is from DMPCampaignDataDictionary.
```