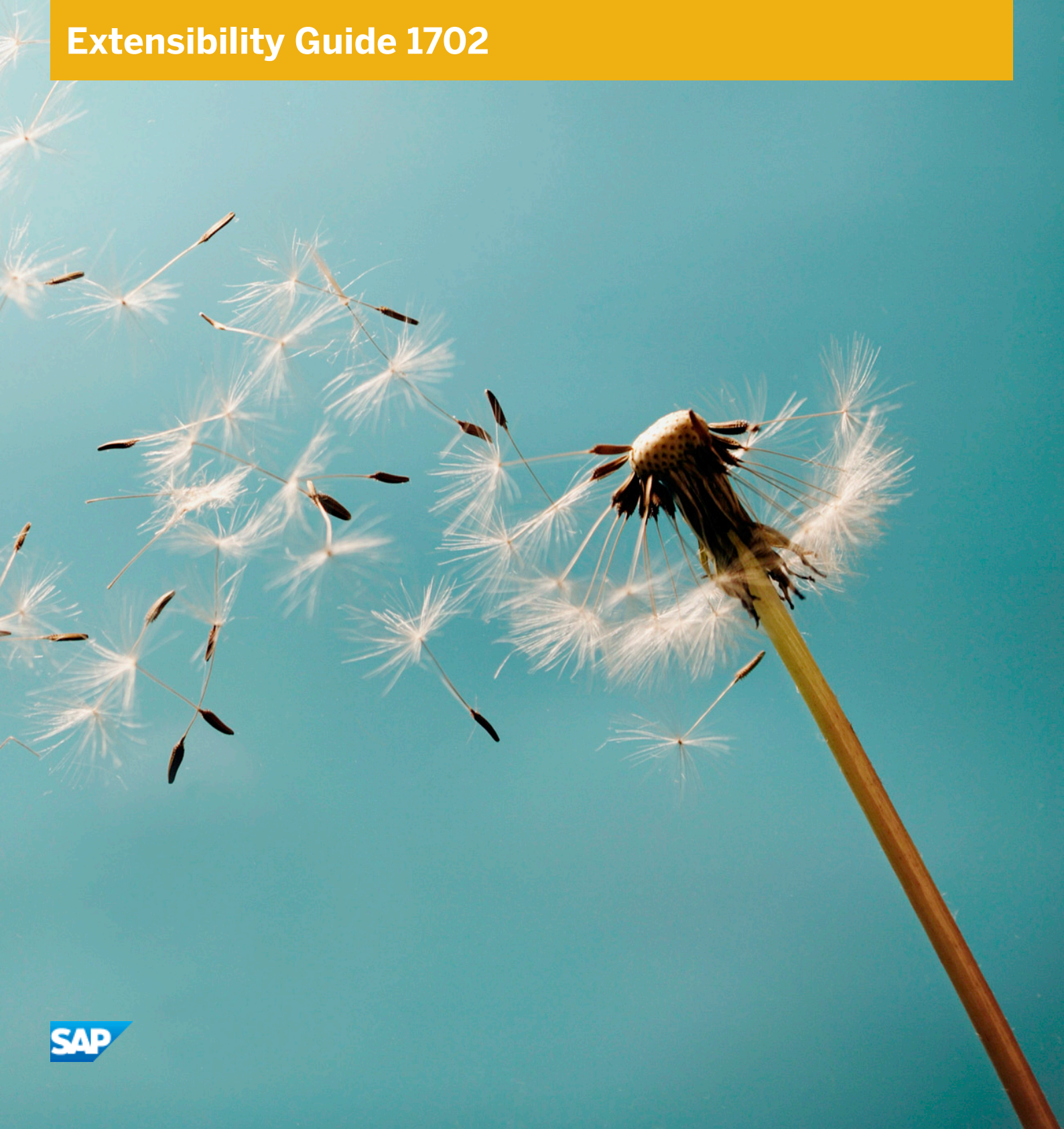


Extensibility Guide 1702



Content

1	Generic Options.	6
1.1	Integration of SAP BusinessObjects Reports.	6
	Identification of OpenDocument Link.	6
	Configuring the Server Connection.	7
	Configuring the Report Launchpad.	7
1.2	Cache Buster Reset.	8
	Enabling the Search for a Data Basis Other Than SAP ERP or SAP CRM.	8
1.3	Including Additional Quick Filter Tiles.	13
2	SAP Hybris Marketing Data Management.	16
2.1	Extending the Data Source and Business Objects.	16
2.2	Additional Applications or Facets.	18
	Including Additional Applications.	18
	Including Additional Facets.	21
2.3	Creating Scores in Account and Contact Profiles.	22
	Creating Heuristic Scores in Predictive Studio.	22
2.4	Setting-Up Predictive Scenarios.	29
	Providing SAP HANA Information Models.	30
	Customizing a Predictive Scenario.	35
	Defining Predictive Scenarios Based on SAP HANA Rules Framework.	37
2.5	Uploading Externally Trained Model Fits.	43
2.6	Enabling Persistence for Predictive Models.	46
2.7	Adding Filter Attributes for Contact Engagement.	47
2.8	Manage Images.	50
	Uploading Images in Bulk.	51
3	SAP Hybris Marketing Insight.	52
3.1	Lead Management.	52
	Extending the Business Partner OData Service in SAP Hybris Marketing.	52
	Extending the Business Document OData Service in SAP Hybris Marketing.	53
	Converting EDMX into XSD Format.	53
	Extending Mapping in SAP NetWeaver Process Integration (PI).	59
	Extending Mapping in SAP HANA Cloud Integration (HCI).	67
	Lead Management.	71
3.2	Relationship Analysis.	71
	Providing SAP HANA Information Models.	72
	Validating Customer Specific Fields in the TransientProvider.	73

	Adapting the Relevant Easy Queries.	74
	Defining Customer Specific Fields as Filter Attributes.	74
	Cleaning up the Cache.	75
	Enabling the Saving of a Personalized View.	75
3.3	Displaying a Key Figure or Characteristic in a Chart.	75
3.4	Integrating SAP Lumira.	78
4	SAP Hybris Marketing Segmentation.	80
4.1	Segmentation.	80
	Providing a Custom Data Source for the Segmentation.	80
	Adapting the Standard Customizing for Segmentation.	82
	Disabling Segment Changes for which a Target Group has been Created.	83
	Defining Additional Preview Types.	83
	Custom Fields in Segmentation.	88
	Setting up the Geospatial Segmentation and Map Preview.	90
	Restricting Data and User Authorization for Segmentation Models.	91
4.2	Target Groups.	92
	Customer-Specific Member Lists.	92
	Enhancing the Target Group Details.	107
	Adding New Fields and Icon Tabs to the Release Target Groups SAP Fiori App.	108
5	SAP Hybris Marketing Recommendation.	110
5.1	Recommendation.	110
	Providing a Custom Data Source or Data Source Pre-Filter.	111
	Providing a Custom Algorithm.	114
	Configuring Data Sources for Algorithms.	116
	Configuring Rule-Based Recommendations.	117
	Consuming Recommendation Models Using Remote Function Call.	127
	Consuming Recommendation Models Using OData.	130
	Providing Impressions Data Using Function Import.	138
	Providing Interactions Data Using Remote Function Call.	139
	Providing Interaction Data Using OData.	142
5.2	Offers.	143
	Custom Fields for Offer Header and Offer Content.	144
	Reporting For Displayed and Clicked Offers Out of External Systems.	148
5.3	Accessing Multichannel Transaction Data.	148
6	SAP Hybris Marketing Planning.	151
6.1	Programs.	151
	Custom Fields in Programs.	151
6.2	Marketing Spend Management.	152
	Adding New Fields to the “My Marketing Spend - Quick Entry” App.	152

	Adding New Fields to the “My Marketing Spend - Details” App.	154
6.3	Calendar.	156
	Customer-Defined Texts.	157
	Customer-Defined Fields and Filter Criteria.	159
	Customer-Defined Widgets.	166
	Customer-Defined Configuration.	174
	Deactivate Personalization.	177
	More Information.	178
7	SAP Hybris Marketing Acquisition.	180
7.1	Creating Customer-Specific Subscription Landing Pages.	180
	Internet Communication Framework.	180
	Background Information.	180
	Logic Flow.	181
7.2	Outbound Integration with SAP CRM.	185
7.3	Extend the Winner Determination for A/B Testing.	185
7.4	Adding Own Actions for Campaign Automation.	186
	Implement Action for Design Time.	188
	Implement Action for Run Time.	190
	Reuse Common Logic by Inheriting.	195
	Reuse Common Logic to Retrieve Personalization Data.	197
	Which Interactions To Create?.	197
	How to Enable a Restart in Case of Technical Failures?.	198
7.5	Custom Fields in Campaign.	198

Document History

Before you start, make sure you have the latest version of this document. You can find the latest version at the following location:

<http://service.sap.com/mkt>

The following table provides an overview of the most important document changes. If the information you are looking for is not described in this guide or if you find something described incorrectly, please send an email to <mailto:saphybrismarketingfeedback@sap.com> and we'll update this guide.

Table 1: Document History

Version	Date	Description
1.0	February 13, 2017 (Release to Customer)	<p>Initial version for SAP Hybris Marketing 1702</p> <p>Major Changes:</p> <p>The following topics have been moved to the Integration Guide:</p> <ul style="list-style-type: none">• Generic Inbound OData Service for Offer Discovery• Import of Offers Using an OData Service• Import of Actual and Committed Spend Using an OData Service• Exporting Target Groups• Importing Brands Using CSV Upload• Importing Custom Dimensions Using CSV Upload• Import of Campaign Execution Plans Using an OData Service

1 Generic Options

This chapter describes cross-component extensibility options.

1.1 Integration of SAP BusinessObjects Reports

You have the possibility to integrate any existing SAP BusinessObjects report such as Web Intelligence reports into your *SAP hybris Marketing* application. In the following section you find a description of the configuration settings which are required to display a SAP BusinessObjects report in the *SAP hybris Marketing* front end.

Note

- The integration of SAP BusinessObject reports in *SAP hybris Marketing* is optional.
- For the integration in *SAP hybris Marketing*, you use SAP BusinessObjects Business Intelligence platform 4.0 SP06 (or higher).

1.1.1 Identification of OpenDocument Link

The integration of a SAP BusinessObject report such as a Web Intelligence report is realized with the help of the OpenDocument interface. Every SAP BusinessObjects report has an OpenDocument link for unique identification in place. With the help you can create a connection to this report from the *SAP hybris Marketing* front end.

You identify the OpenDocument URL of the report you require to integrate by following the instructions of section 6.11, *To create an OpenDocument link for an object*, in the corresponding user guide on the SAP Help Portal at:

<http://help.sap.com/bobip>  *End User Guides – Application Help*  *User Guides*  *BI Launch Pad User Guide 4.1* 

(http://help.sap.com/businessobject/product_guides/sbo41/en/sbo41_bip_bilaunchpad_en.pdf )

Once you identified the OpenDocument link, keep it for further processing in the following configuration steps.

Note

A correct OpenDocument URL includes the string `/BOE/OpenDocument!`

1.1.2 Configuring the Server Connection

SAP BusinessObjects reports are located on the server of the SAP BusinessObjects Business Intelligence platform. To retrieve the report from the [SAP hybris Marketing](#) front end, you need to create an RFC connection between the AS ABAP system and the server.

To set up this RFC connection, you do the following:

1. Logon to your AS ABAP system and enter TA SM59.
2. Select [HTTP Connections to ABAP System](#) and choose the [Create](#) pushbutton.
3. Enter **BOE_SERVER** as RFC destination.
4. On the [Technical Settings](#) tab, enter the host of the server on which the required report is located, in field [Target Host](#) with fully qualified domain.
5. In field [Service No.](#), you enter the port of the server. The port is part of the OpenDocument URL, you identified in the previous step.
6. Enter **/BOE** in field [Path Prefix](#).
7. On the [Logon & Security](#) tab, set the [SSL](#) radio button in the [Security Options](#) section to [Active](#).
8. Save your entries.

1.1.3 Configuring the Report Launchpad

While you created a general connection between SAP BusinessObjects server and AS ABAP system in the previous step, you now have to be more precise, and define a connection to the required report in the report launchpad of the AS ABAP system (transaction `LPD_CUST`).

To do so, proceed as follows:

1. Logon to your AS ABAP system and enter transaction `LPD_CUST`.
2. Search for the entry with role [HPA](#) and Instance [CUAN](#), select it, and choose the [Change](#) pushbutton.
3. Choose the [New Application](#) pushbutton, and enter any connection name in field [Link Text](#) in the [Link Details](#) section.
4. Enter **URL** in field [Application Type](#).
5. Choose the [URL Editor](#) pushbutton next to the [URL](#) field in the [Application Parameter](#) section.
6. Enter the OpenDocument URL you identified previously, but delete all characters before [OpenDocument](#). This means, the entered URL must start with **/OpenDocument**!
7. Enter **BOE_SERVER** in field [System Alias](#). This alias links to the RFC destination of the RFC connection created in the previous step, so that the system can create a complete URL at runtime.
8. Choose the [Show Advanced \(Optional\) Parameters](#) pushbutton, choose the [Editor Application Alias](#) pushbutton next to the [Application Alias](#) field in the [Application-Related Parameters](#) section, and enter any application alias. Keep this alias for further processing.
9. If you want to use your SAP BusinessObjects reports in an item details (Thing Inspector) facet for a specific object (for example, account, campaign), we recommend you define your own target application parameters. To do so, proceed as follows:
 - If your SAP BusinessObjects report uses a prompt value for filtering, you can configure which field should be used at runtime as source to fill this parameter. Field [sObjectId](#) is filled by item details for account, campaign and target group. This field contains the ID of the selected object.

- For naming of the prompt parameter, see *Input Parameters*, and *IsS (NAME)* at:
<http://help.sap.com> ► *Analytics* ► *Business Intelligence* ► *Business Intelligence Platform (Enterprise)* ► *SAP BusinessObjects Business Intelligence Platform 4.1* ► *Development Information* ► *Viewing Documents Using OpenDocument 4.1* ►
(http://help.sap.com/businessobject/product_guides/sbo41/en/sbo41_opendocument_en.pdf)

10. Enter **BOE** in field Additional Information and save your entries.

1.2 Cache Buster Reset

If the Cache Buster is not reset after a change in the application code, for example, after an ABAP import or the installation of a note, the user might receive the message *Http status code 500 — Requested Resource outdated*. You can fix this either by waiting (2 hours maximum) for the Cache Buster to be reset automatically, or by running the program `/UI5/RESET_CACHEBUSTER`. The program updates the information on the server that is checked by the client in order to decide, whether a resource can be read from the browser cache or whether it has to be retrieved from the server.

If changes are frequently being made to the application code during regular working hours, you can either run the report as required after every ABAP import, or schedule it to run as a periodic background job.

1.2.1 Enabling the Search for a Data Basis Other Than SAP ERP or SAP CRM

If you want to run *SAP hybris Marketing* on a data basis other than SAP ERP or SAP CRM, you have to enhance the existing standard SAP HANA information models and the corresponding queries to support the application's search functionality. Without adapting information models and queries, you can neither use the quick search in the *Home* workset nor the search fields of the list views in the *Target Groups* and *Initiatives* worksets, and the *Accounts* subworkset.

You can create your own information models and queries or you can copy and adapt the existing standard SAP ERP or SAP CRM models and queries. In addition, you need to create or copy and adapt the corresponding result structures for the queries. It is important that you use exactly the same field names that exist in the SAP ERP or SAP CRM structure.

The adjustment of the following three standard queries and the underlying information models in the customer namespace is mandatory:

- `QUICK_SEARCH` for the search in the *Home* workset
- `SEARCH` for the search in the list views of *Target Groups*, *Initiatives*, *Accounts*
- `READ_CUST_ID_FOR_AUTH_CHECK` for the authorization check if you want to support the search in *My Accounts* in the *Accounts* subworkset on the user interface.

1.2.1.1 Creating Customer-Specific SAP HANA Information Models for the Search

To ensure the full search capability of *SAP hybris Marketing*, you must enhance the existing standard SAP HANA information models that are relevant for the search. In the SAP HANA modeler, you can either create your own information models or copy and adapt the existing standard models.

You can copy and adapt the following attribute views in the SAP HANA modeler at `sap.hana-app.cuan.common.crm.internal`:

- `AT_CUSTOMER_SEARCH` for the quick search in the *Home* workset and the search in the list views
- `AT_CUSTOMER_BY_TEAM_MEMBER` for the authorization check and search in *My Accounts* in the *Accounts* subworkset

You can create one or more information models for the search depending on your requirements. You must only ensure that the following mandatory fields are included in your information models:

- `KUNNR` (mandatory for all queries)
- `COUNTRY` (mandatory for all queries)
- `NAME1` (mandatory for quick search)
- `CITY1` (mandatory for quick search)
- `BRSCH.description` (mandatory for quick search)
- `BNAME` (mandatory if you want to enable the search in *My Accounts* on the user interface)

Caution

Ensure that the field names you use in your own models are exactly the same as in the standard information model such as `KUNNR` for the account ID or `COUNTRY.description` for the country text field.

1.2.1.2 Creating a Customer-Specific Result Structure for the Search

As you have to create customer-specific search queries if you use a data basis other than SAP ERP or SAP CRM, you must create the corresponding result structure first.

For the customer-specific query that you create to substitute the standard query `QUICK_SEARCH`, you must use the existing standard result structure `CUAN_S_QR_CUST_RT_QUICK_SEARCH`. You do not need to create an own result structure.

For the customer-specific query that you create to substitute the standard query `READ_CUST_ID_FOR_AUTH_CHECK`, you must use the existing standard result structure `CUAN_S_KUNNR`. You do not need to create an own result structure.

But for the customer-specific query `SEARCH`, you can create your own result structure as the standard result structure does not cover your requirements.

To do so, you do the following:

1. Choose transaction `SE11` and create a data structure in the customer namespace.

- Enter any required name in the *Short Description* field and enter the required fields under the *Components* tab. You can copy the fields from the standard structure CUAN_S_QR_CUST_ROOT_ELEMENTS. You must ensure that the following mandatory fields are included in your result structure:
 - KUNNR (mandatory for all queries)
 - NAME1 (mandatory for quick search)
 - CITY1 (mandatory for quick search)
 - BRSCH_T (mandatory for quick search)
 - COUNTRY (mandatory for authorization check)
 - BNAME (mandatory if you want to enable the search in *My Accounts* on the user interface)
- Create an appropriate table type for your result structure.

⚠ Caution

Ensure that the field names you use in your own structure are exactly the same as in the standard structure such as KUNNR for the account ID or COUNTRY_T for the country text field.

Structure

ZMY_S_CUSTOMER_RESULT

Active

Short Description

My Result Structure for Customer Query

Attributes

Components

Entry help/check

Currency/quantity fields

Predefined Type

1 / 12

Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
KUNNR	Types	CUAN_CUSTOMERID...	CHAR	10	0	Account ID With Alpha Conversion
BRSCH	Types	CUAN_BRSCH	CHAR	4	0	Industry Key
BRSCH_I	Types	CUAN_BRSCH_I	CHAR	20	0	Description of Industry Key
NAME1	Types	AD_NAME1	CHAR	40	0	Name 1
COUNTRY	Types	LAND1	CHAR	3	0	Country Key
COUNTRY_I	Types	LANDX	CHAR	15	0	Country Name
CITY1	Types	AD_CITY1	CHAR	40	0	City
POST_CODE1	Types	AD_PSTCD1	CHAR	10	0	City postal code
REGION	Types	REGIO	CHAR	3	0	Region (State, Province, County)
REGION_I	Types	BEZEI20	CHAR	20	0	Description
STREET	Types	AD_STREET	CHAR	60	0	Street
HOUSE_NUM1	Types	AD_HSNM1	CHAR	10	0	House Number

Figure 1: Customer-Specific Result Structure with Example Fields

1.2.1.3 Creating Enhancement Business Object for the Search

To be able to create customer-specific search queries, you must enhance the corresponding standard business object (BO) `CUAN_CUSTOMER` at which the standard queries are located. If you have already enhanced this BO due to a previous extension of *SAP hybris Marketing*, you can skip this step.

To create a BO enhancement, you do the following:

1. Choose transaction BOB.
2. Choose the *Business Object Enhancement* pushbutton. The *Create Enhancement* wizard appears.

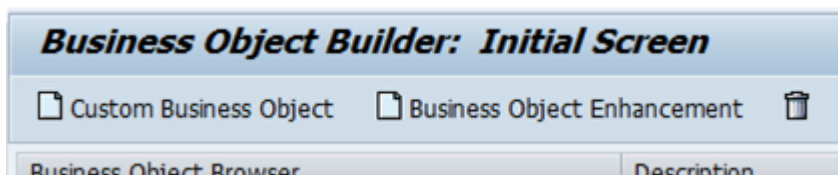


Figure 2: Business Object Enhancement Pushbutton

3. Choose *Continue* and enter `CUAN_CUSTOMER` in the *Base Business Object* field.
4. Enter a technical name in the customer namespace (beginning with **z**) in the *Enhancement Name* field.
5. Enter any required description in the *Description* field.
6. You can leave the *Namespace* field empty but you have to specify **z** in the *Prefix* field.

Base Business Object	CUAN_CUSTOMER
Enhancement Name	Z_MY_CUSTOMER
Description	My Customer Enhancements
Namespace	
Prefix	z

Figure 3: Attributes for the Enhancement BO

7. Choose [Continue](#).
8. In the *Constants Interface* field, specify an entry such as **ZIF_MY_CUSTOMER_C**.

Technical Name	Z_MY_CUSTOMER
Constants Interface	ZIF_MY_CUSTOMER_C
	Display Name

Figure 4: Example for the Constants Interface

9. Choose [Continue](#) and decide whether it should be possible to enhance your enhancement itself.
10. Choose [Continue](#) and then [Complete](#).

Your enhancement BO is created.

1.2.1.4 Creating Customer-Specific Queries for the Search

Having created an enhancement BO for the standard BO CUAN_CUSTOMER in the previous step, you can now create the three required customer-specific search queries that substitute the standard queries SEARCH, QUICK_SEARCH, and READ_CUST_ID_FOR_AUTH_CHECK.

To do so, you do the following:

1. Choose transaction BOB, select the enhancement BO that you have created in the previous step and choose [Open](#) in the context menu.
2. Select the *ROOT* node in the *Node Browser* section and choose [Create Query](#) in the context menu.
3. Choose [Continue](#) in the appearing *Create Query* wizard.
4. Enter the required query name in the *Query Name* field. Follow the query naming convention and start the name with **_Z_**. For the customer-specific SEARCH query, enter **_Z_SEARCH**, for example.

Note

If the system does not accept a query name beginning with `_Z_`, enter `Z_<query_name>` and change the name to `_Z_` in the *Entity Browser* section once the query is created.

Enter any required query description in the *Description* field and choose *Continue*.

5. In the next step, select the *Custom Query* radio button and choose *Continue*.
6. In the next step, enter the following parameters for the customer-specific `SEARCH` query:
 - *Implementing Class*: `CL_CUAN_READ_ACCESS_COMMON`
 - *Data Type*: `CUAN_S_Q_CUST_ROOT_ELEMENTS`
 - *Result Type*: Enter the name of the result type that you have created in section 11.2, *Creating a Customer-Specific Result Structure for the Search*.
 - *Result Table Type*: Enter the name of the result table type that you have created in section 11.2, *Creating a Customer-Specific Result Structure for the Search*.
7. Choose *Continue*.
8. In the next step, enter the path to the corresponding SAP HANA information model that you have created in section 11.1, *Creating Separate SAP HANA Information Models for the Search*. The path must be entered in the following format: `"_SYS_BIC"."<your_path_in_SAP_HANA_Modeler>/<name_of_your_information_model>"`
Example: `"_SYS_BIC"."sap.hana-app.cuan.common.internal/AT_CUSTOMER_SEARCH"`
9. Choose *Complete* to finally create your search query.
10. Repeat step 1 to step 9 to create a customer-specific query as a substitute for the standard query `QUICK_SEARCH`. Replace the parameters in step 6 with the following:
 - *Implementing Class*: `CL_CUAN_QS_CUSTOMER`
 - *Data Type*: `HPA_S_Q_QUICK_SEARCH` (as in the standard query)
 - *Result Type*: `CUAN_S_QR_CUST_RT_QUICK_SEARCH` (as in the standard query)
 - *Result Table Type*: `CUAN_T_QR_CUST_RT_QUICK_SEARCH` (as in the standard query)
11. Repeat step 1 to step 9 to create a customer-specific query as a substitute for the standard query `READ_CUST_ID_FOR_AUTH_CHECK` (if you want to enable the search in *My Accounts* on the user interface). Replace the parameters in step 6 with the following:
 - *Implementing Class*: `CL_CUAN_READ_ACCESS_COMMON`
 - *Data Type*: `CUAN_S_Q_CUST_ROOT_AUTH` (as in the standard query)
 - *Result Type*: `CUAN_S_KUNNR` (as in the standard query)
 - *Result Table Type*: `CUAN_T_KUNNR` (as in the standard query)

1.3 Including Additional Quick Filter Tiles

Use

You can include additional quick filter tiles in the *Target Group* workset and in the *Initiatives* workset. You perform the following main tasks to include a new quick filter tile:

- Adapt the SAP HANA information models that provide the counts for the quick filter tiles

- Adapt the relevant data dictionary structure
- Configure the new quick filter tile in the Customizing for SAP Hybris Marketing
- Implement a BAdI for filter definition and counter determination

Procedure

Adapting the Relevant SAP HANA Information Models

The counts for the quick filter tiles are provided by calculations views. In the calculation view, you specify additional count logic for the new quick filter tile you want to include. The following calculation views are used:

- Counts for Quick Filter Tiles (All Initiatives) (CA_CUAN_INITIATIVE_OWL_QF_ALL).
The calculations view provides the counts for all initiatives.
- Counts for Quick Filter Tiles (My Initiatives) (CA_CUAN_INITIATIVE_OWL_QF_MY).
The calculations view provides the counts only for the initiatives for which the user is responsible.
- Counts for Quick Filter Tiles (Target Groups) (CA_CUAN_TG_QF_COUNTER).
The calculations view provides the counts for all target groups, and also the counts for the target groups for which the user is responsible.

Note

For the quick filter tiles in the *Initiatives* workset, you adapt both calculation views, the one for all for all initiatives, and the one only for the initiatives for which the user is responsible. For the quick filter tiles in the *Target Group* workset, you adapt only one calculation view (as it provides the both counts).

For more information, see the *SAP HANA Developer Guide* at <http://help.sap.com/hana>  *SAP HANA Appliance Software* .

To adapt the relevant calculation view(s) in SAP HANA Studio, proceed as follows:

1. Define the *Output Parameter* for the new counter.
2. Insert the section for the new counter in the SQL statement of the calculation view(s).
3. Save your changes.

Enhancing the Relevant Dictionary Structure

Use the *ABAP Dictionary Maintenance* (transaction SE11) to enhance the following dictionary structures, depending on the workset where you include the new quick filter tiles:

- Quick Filter: Counter for Tiles (CUANC_S_QF_COUNTER)
The dictionary structure is relevant for the quick filter tiles in the *Initiatives* workset.
- Target Group OWL Tile Counter (CUAN_S_QR_TG_QF_COUNTER)
The dictionary structure is relevant for the quick filter tiles in the *Target Groups* workset.

In the relevant dictionary structure, under *Component*, add the output parameter you have defined for the new counter in the calculation view(s).

i Note

In the dictionary structure for the quick filter tiles in the *Target Groups* workset (CUAN_S_QR_TG_QF_COUNTER), complete the output parameter entry adding _SUM (the new entry should read: <OUTPUTPARAMETER>_SUM).

Configuring a New Quick Filter Tile

You configure additional quick filter tiles in Customizing for *SAP hybris Marketing* under ► *General Settings* ► *Set Up Quick Filter Tiles* ► as follows:

1. Choose *New Entries* to add a new quick filter tile to a workset (choosing the according the *List ID*).
2. Enter a *Tile ID* for the new quick filter tile. Only integers are allowed.

i Note

You require the tile ID when you implement a BAdI, which is the final step you perform to include new quick filter tiles.

3. Complete the Customizing for the new quick filter tile. For more information, see the help document for the *Set Up Quick Filter Tiles* customizing activity.

Implementing the BAdI

As the final step of including new quick filter tiles, you implement a BAdI that provides methods for the definition of filters for the list of target groups or initiatives, and for the determination of the counters. You use different methods for each of the two worksets where you can include new quick filter tiles. You can access the BAdI in Customizing for *SAP hybris Marketing* under ► *General Settings* ► *BAdI: Definition of Filter Options for Quick Filter Tiles* ►. For more information, see the help document for the customizing activity.

2 SAP Hybris Marketing Data Management

2.1 Extending the Data Source and Business Objects

You can extend the SAP HANA data source and the according Business Object (BO) to adapt SAP Hybris Marketing according to your requirements. To adapt the data structure of the application you perform the following tasks:

- Copy the relevant SAP HANA information model(s), and adapt the model(s), for example, to include additional attributes. Clear system buffers.
- Extend the Business Object data structure (ABAP backend), to include, for example, additional attributes in the application.

Copying SAP HANA Informations Models

Use the *Mass Copy* function of SAP HANA Studio to copy the relevant SAP HANA information models:




1. Start SAP HANA Studio. On the *Quick Launch* tab, under *Content*, choose *Mass Copy*.
2. In the *Copy Models* dialog, you first configure the package mapping. Choose *Add*. Use the input help of the *Target* column to select the (top-level) target folder for your copies of the relevant SAP HANA information models. If necessary, create a custom folder before you configure the package mapping.

Note

All copies of the SAP HANA information models you want to extend must reside in the same custom (top-level) folder.

3. Having configured the package mapping in the *Copy Models* dialog, choose *Next*. In the *Available* panel, select the relevant SAP HANA information models (or a package). Choose *Add*. To check the summary of selected models choose *Next*.

Note



For more information about the SAP HANA information models that are relevant for specific worksets, subworksets or KPIs of SAP Hybris Marketing, see the business content documentation at <http://help.sap.com/mkt>  *Configuration and Deployment Information*  *Business Content* .

4. In the *Summary of the Models* of the *Copy Models* dialog, check the properties of the listed models. By default, all models you have added are selected for the mass copy. Deselect models you do not want to include. Once you are ready with the selection, choose *Finish* to run the mass copy.





Note

Using the mass copy function (including the package mapping) you enable the system to refer to your custom SAP HANA information models instead referring to the SAP delivered standard models.

Use your custom copies of the relevant SAP HANA information models (see the folder you have specified in the package mapping) to extend the data source according to your requirements.

For more information about how to extend SAP HANA information models, see the [SAP HANA Modeling Guide](http://help.sap.com/hana_platform) at http://help.sap.com/hana_platform  [Modeling Information](#) . Also, see the [SAP HANA Developer Guide](#), which is available at the same location under [Development Information](#).











Clearing Buffers

To make the referring to your custom SAP HANA information models effective you need to clear a set of system buffers, such as the buffer of the Business Object Processing Framework (BOPF). Use Customizing for [SAP Hybris Marketing](#) (formerly SAP Customer Engagement Intelligence) under  [General Settings](#)  [Extensibility](#)  [Clear BOPF Metadata from Buffer](#)  to reset the buffers.

Extending the Data Structure in a Business Object

In addition to adapting a custom copy of the relevant SAP HANA information model, you need to extend the relevant BO data structure. Use the [SAP Business Objects in the Business Objects Builder](#) (transaction BOB) to extend a BO data structure as follows:

1. In the [Business Object Browser](#), open the [SAP Business Objects](#) folder.
2. Double click the relevant BO, for example, [Interaction Contact](#) (CUAN_INTERACTION_CONTACT) to open the BO.
3. In the [Node Browser](#), double click the BO node that you want to extend to display the relevant [Extension Includes](#). You can only extend a BO node if an extension include is entered.
4. Under [Extension Includes](#), double click the relevant include, for example, the [Persistent Include](#) INCL_EEW_CUAN_CE_IC_ROOT. As a result, the ABAP dictionary for the selected include is displayed.
5. In the ABAP dictionary, choose [Append Structure ...](#) to extend the include structure. Provide a name for the append structure using a naming convention that indicates the customer namespace, for example, beginning the name with the appendix [Z_](#).

For more information about how to use append structures, see the [ABAP Dictionary](#) documentation at <http://help.sap.com/nw75>  [SAP NetWeaver Library: Function-Oriented View](#)  [Application Server](#)  [Application Server ABAP](#)  [Application Development on AS ABAP](#)  [ABAP Programming Tools](#)  [ABAP Dictionary](#)  [Tables](#)  [Making Changes to Tables](#)  [Adding an Append Structure](#) .

6. Use the newly created append structure to include, for example, additional attributes (using each attribute's name, data type, and field length as defined in the adapted SAP HANA information model).

To do so, proceed as follows:

1. Go to transaction HPA_WHERE_USED_INCL and enter the name of the include structure that you have extended in the previous step, in the [Name of Include](#) field and choose the [Execute](#) pushbutton.

2. As a result, you are provided with a list of involved SAP HANA information models. You must add the new field to the output structure of each listed information model in the SAP HANA studio.

- If a listed information model still resides in the SAP namespace, you have to copy it first. For information about how to copy information models, see the *Copying SAP HANA Informations Models* section above.
- If a listed information model resides already in the customer namespace (which means, is already copied), you can directly edit this very model.

7. Use transaction `HPA_CLEAR_BUFFERS` (also available in Customizing for *SAP hybris Marketing* (formerly SAP Customer Engagement Intelligence) under **General Settings** **Extensibility** **Clear BOPF Metadata from Buffer**).

In addition, the transaction clears the SAP NetWeaver Gateway shared memory which ensures that the new field is taken over to the entity within the OData service.

In SAP Hybris Marketing, you can include additional applications, and facets. There are alternatives for the integration of additional applications or facets. Consider the following:

In both cases, make sure to enable Single Sign On (SSO) to avoid an additional user log on. You can enable SSO, for example, using certificates.

For any additional application you want to include in *SAP hybris Marketing*, you create a specific PFCG role, create a new launchpad entry and specify the application type details.

- SAP BusinessObjects reports, such as WebIntelligence, Dashboards, BOE server links to report lists
- Applications based on SAPUI5
- WebDynpro applications

Creating the Required PFCG Role

Perform the following steps to define the PFCG role:

1. In your AS ABAP system, use the *Role Maintenance* (transaction PFCG) to create a new role based on the standard role *SAP hybris Marketing: General Access* (SAP_CUSTOMER_ENG_INTELLIGENCE).
 - For more information about the standard roles that are provided with *SAP hybris Marketing*, see *Roles in SAP hybris Marketing* on the help portal at <http://help.sap.com> ► *SAP Business Suite* ► *SAP HANA Innovations for SAP Business Suite* ► *Applications powered by SAP HANA* ► *SAP hybris Marketing, Powered by SAP HANA* ► *SAP hybris Marketing User Management* ►.
 - For more information about how to change standard roles, see *Changing Standard Roles* on the help portal at <http://help.sap.com> ► *SAP NetWeaver* ► *SAP NetWeaver 7.5* ► *Application Help* ► *Function-Oriented View* ► *Security* ► *Identity Management* ► *User and Role Administration of Application Server ABAP* ► *Configuration of User and Role Administration* ► *Role Administration* ► *Role Administration Functions* ►. Here, you can also find information about the *Role Menu* under ► *Role Administration* ► *Creating Single Roles* ►.
2. In the *Role Menu*, expand the folder structure under ► *High Performance Applications* ► *hybris Marketing* ►. The folder structure defines the worksets and sub worksets of *SAP hybris Marketing*. Determine the structure node where you want to include the additional application.

⚠ Caution

In the folder structure of the *Role Menu* do not change or delete the nodes *High Performance Applications* (HPA), *hybris Marketing* (CUAN), and the Thing Inspectors of *hybris Marketing*.

3. Select the folder structure node and choose *Create Folder*. Specify a name for the new node. This name is used for the additional application on the UI of *SAP hybris Marketing*.
4. Choose *Other Node Details*. Enter the *Application Alias* for the additional application. Note that the application alias is used as a reference in the launchpad of *SAP hybris Marketing*. Make sure to define a unique alias. In addition, we recommend defining a proper name since the alias is displayed in the URL when the application is called from the UI.
5. Save the new role you have created.

Creating the Required Launchpad Entry

Perform the following steps for the definition of the launchpad entry:

1. In the *Overview of Launchpads* (transaction LPD_CUST), select the role *HPA* (instance *CUAN*). Choose *Change*.
For more information about how to use the *Overview of Launchpads*, see the help portal at <http://help.sap.com/netweaver> ► *SAP NetWeaver 7.5* ► *Application Help* ► *Function-oriented view* ► *Application Server* ► *Application Server ABAP* ► *UI Technologies in ABAP* ► *Launchpads* ► *Working with Launchpads at Design Time* ►.

i Note

See the section about *Re-Displaying an SAP-Delivered Launchpad* when you want to adapt a custom launchpad according to an enhanced standard launchpad delivered by SAP, for example, with a support package.

2. Choose *New Application*. For the *Link Text*, enter a descriptive name for the new application. The name is used in the launchpad's application list for the *SAP hybris Marketing* shell.
3. Select an *Application Type*, for example, *URL*.
4. Use one of the following sections to complete the process, depending on the type of application you are including.

Application Type: URL

This application type is used for SAPUI5 based applications, and for SAP BusinessObjects applications. You specify the launch of SAP BusinessObjects applications as described in the Extensibility Guide section *Configuration of Report Launchpad*. Make sure to specify the SAP BusinessObjects application entering **BOE** under *Additional Information* (see step 4 in the procedure for SAPUI5 based applications).

For SAPUI5 based applications, perform the following steps:

1. Under *Application Parameter*, choose *URL Editor*. Enter the complete system path where the application you want to include can be accessed.

Use the following format for the path information: `/<name of the UI5 application>/<name of application folder>` as found under **Page Fragments > WebContent**. For example, enter `/cuan_ai_ra_wsi/cuan_ai_ra_wsi` for the path of the *Relationship Analysis* application (CUAN_AI_RA_WSI).

Note

Make sure to activate the ICF node that corresponds to the name of the UI5 application. Use transaction *HTTP Service Hierarchy Maintenance* (SICF) to activate the ICF service for the UI5 application.

2. Under *System*, select *HTML5APPS* for the *System Alias*.
3. Choose *Show Advanced (Optional) Parameters*.
4. Under *Application-Related Parameters*, enter the following:
 - The *Application Alias* you have defined in the role menu
 - For *Additional Information* enter the path to the Java Script view of your SAPUI5 application. Use the following format for the path information: `<name of the UI5 application>;<path of Java Script view>`. The path for the Java Script view is built as follows: `<name of application folder under Page Fragments > WebContent >><name of Java Script view, without .view.js>`. For example, enter `cuan_ai_ra_wsi;cuan_ai_ra_wsi.relana` for the *Relationship Analysis* application.
5. The required *Portal Parameters* are provided by the system.
6. Save the new application launchpad entry you have defined.

Application Type: WebDynpro ABAP

For the launch of WebDynpro applications perform the following steps:

1. Under *Application Parameters*, enter **SAP** for the *Name Space*.
For *Application*, enter the name of the WebDynpro application you want to include.
2. Under *System*, enter **SAP_LocalSystem (100)** for the *System Alias*. In addition, select *Force local system if NWBC*.
3. Choose *Show Advanced (Optional) Parameters*.
4. Under *Application-Related Parameters*, enter the following:
 - The *Application Alias* you have defined in the role menu
 - The *Configuration* ID of the WebDynpro application
5. The required *Portal Parameters* are provided by the system.
6. Save the new application launchpad entry you have defined.

2.2.2 Including Additional Facets

For any additional facet you want to include in *SAP hybris Marketing*, you create a specific PFCG role, create a new launchpad entry and specify the application type details.

Creating the Required PFCG role

Perform the following steps to define the PFCG role:

1. In your AS ABAP system, use the *Role Maintenance* (transaction PFCG) to create a new role based on the standard role *SAP hybris Marketing: General Access* (SAP_CUSTOMER_ENG_INTELLIGENCE).
For more information on the standard roles that are provided with SAP Hybris Marketing, see *Roles in SAP hybris Marketing* on the help portal at ► <http://help.sap.com> ► *SAP Business Suite* ► *SAP HANA Innovations for SAP Business Suite* ► *Applications powered by SAP HANA* ► *SAP hybris Marketing, Powered by SAP HANA* ► *SAP hybris Marketing User Management* ►.
For more information on how to change standard roles, see *Changing Standard Roles* on the help portal at ► <http://help.sap.com> ► *SAP NetWeaver* ► *SAP NetWeaver 7.5* ► *Application Help* ► *Function-Oriented View* ► *Security* ► *Identity Management* ► *User and Role Administration of Application Server ABAP* ► *Configuration of User and Role Administration* ► *Role Administration* ► *Role Administration Functions* ►.
Here, you can also find information about the *Role Menu* under ► *Role Administration* ► *Creating Single Roles* ►.
2. In the *Role Menu*, expand the folder structure under ► *High Performance Applications* ► *Thing Inspectors of hybris Marketing* ►. The folder structure defines the item details (Thing Inspectors) for *Account*, *Target Group*, and *Initiative*. Each folder contains the facets for the item details of *SAP hybris Marketing*, for example, *Members* and *Initiatives* for the target group item details. Determine the structure node where you want to change the list of facets to be displayed, by either adding new facets or removing facets. Here, you can also define and change the sequence of the facets on the user interface (UI).

Caution

In the folder structure of the *Role Menu*, do not change or delete the nodes *High Performance Applications* (HPA), *hybris Marketing* (CUAN), or *Thing Inspectors of hybris Marketing*.

3. Select the folder structure node and choose *Create Folder*. Specify a name for the new node. This name is used for the additional facet in the item details on the UI of *SAP hybris Marketing*.
4. Choose *Other Node Details*. Enter the *Application Alias* for the additional facet.

Note

The application alias is used as a reference in the launchpad of *SAP hybris Marketing*. Make sure you define a unique alias.

5. Save the new role you have created.

Creating the Required Launchpad Entry

For instructions on how to create the required launchpad entry, see **Creating the Required Launchpad Entry** in the chapter *Including Additional Applications*.

Note

You can also integrate your own BusinessObjects reports in new facets. To do so, see the installation guide for *hybris Marketing* under <http://service.sap.com/instguides> ► *SAP In-Memory Computing* ► *SAP hybris Marketing 1.0* ► in the chapter *Configuration of Report Launchpad*.

2.3 Creating Scores in Account and Contact Profiles

You can create your own scores and display them on the profile of an individual account, contact, or consumer on the user interface of SAP Hybris Marketing. Scores can be calculated in two different ways:

- A *Predictive Score* is calculated based on a predictive model, typically using statistical methods, such as logistic regression. For example, buying propensity or lead propensity are predictive scores.
- A *Heuristic Score* is calculated based on manually defined scoring rules. It is called heuristic as manually defined rules are based on experience and educated guess. For example, you define a rule that classifies a customer as churned when he did not order for six months.

Heuristic scores can be created using both *Score Builder* or *Predictive Studio*.

- In *Score Builder*, the created heuristic score is based on a segmentation profile. You can select from a pre-configured set of HRF vocabulary in a drop down menu to define your rule sets. For more information, see the documentation on the *Score Builder* in the application help at <http://help.sap.com/mkt>.
- In *Predictive Studio* the created heuristic score is based on SAP HANA information models. You can create your own HRF vocabulary and then use it to define your rules. For more information on creating a HRF vocabulary, see [Defining Predictive Scenarios Based on SAP HANA Rules Framework](#). [page 37]

Read on for more information on how to create heuristic scores in *Predictive Studio*.

2.3.1 Creating Heuristic Scores in Predictive Studio

The process of creating a heuristic score in *Predictive Studio* includes the following steps:

- **Creating SAP HANA Information Models in the SAP HANA Studio**

You must create information models that build the basis for your heuristic score as they return the required calculation results. The models must suit the requirements of the heuristic score that you want to display on the user interface.

Caution

Make sure that all information models that you create are cross-client to ensure proper data flow.

You are provided with information models that serve as templates for your models. You can copy and adapt these templates according to your requirements.

Note

Ensure that the interdependencies between the information models are kept after copying the template models. If, for example, a procedure is called from within a script-based calculation view, then you have to replace the corresponding path and, if applicable, the name.

It is not allowed to change the input and output parameters of the information models.

- **Executing Customizing Activities in Customizing for Segmentation**

Customizing for Segmentation is required for two reasons:

- Customizing for Segmentation is required if you want to use the heuristic scores that you create in Segmentation itself.
- Segmentation Customizing provides features for performance improvements. Some of the views in Segmentation Customizing activities can be used as data provider with known field names. Thus, Segmentation Customizing is necessary to integrate scope and improving performance while displaying scores in profiles.

- **Executing Customizing Activities in Customizing for Predictive Scenarios**

The creation process of heuristic scores is embedded into the predictive scenario framework/environment. Each heuristic score corresponds to a predictive scenario that is based on a standard heuristic model.

For this reason, you have to carry out a few Customizing activities in Customizing for Predictive Scenarios during the creation process of your score.

For heuristic models (in difference to predictive models) it is not necessary to define a predictive data source (that is a SAP HANA information model used as a data source for the training of a predictive model) as the model training step is not relevant for heuristic scores but for predictive scores only.

For more information about Predictive Model Management, see the SAP Help Portal at:

<http://help.sap.com/mkt>  *Application Help*  *SAP Hybris Marketing Worksets and Applications*  *Recommendation*  *Predictive Studio* 

http://help.sap.com/saphelp_mkt118/helpdata/en/21/3f155201049a33e100000000a44538d/content.htm?frameset 

- **Maintaining the Heuristic Model on the User Interface for Predictive Models**

As a final step, you must manually create the heuristic model for you heuristic score in the *Predictive Studio* app on the user interface of *SAP Hybris Marketing*.

The predictive model allows you to do the following:

- Specify the implementation method for the score (as it is possible that several slightly different formulas exist)
- Define restrictions for the score calculation (for example, one or more countries for which the score will be valid)
- Display a preview of the score calculation results
- Release the score to make it available on a profile or in Segmentation

2.3.1.1 Creating SAP HANA Information Models in the SAP HANA Studio

To create a suitable SAP HANA information model for your required heuristic score, you are provided with template information models for the following example score:

Age Score (a score for a contact that calculates the contact's age)

To create your own heuristic score, you must copy and adapt the following template information models for the *Age Score*:

- Calculation view `sap.hana-app.cuan.contact.score/CA_CE_CONTACT_AGE_SCORE_EXT` (graphical information model providing the result of the model execution to the consuming application and referring to the corresponding script-based information model)

Caution

You can use this information model for not more than exactly one self-defined score.

- Calculation view `sap.hana-app.cuan.contact.score.internal/CA_CE_CONTACT_AGE_SCORE` (script-based information model holding the scoring of customers and reads Customizing entries such as the actual score value (the age in this example score, which can be divided into different ranks such as "young", "best age", or "mature", for example))
- Procedure `sap.hana-app.cuan.contact.score.internal/PR_CE_CONTACT_AGE` (procedure containing the formulas/select statements to calculate the score)
In this procedure, a table must be associated, in which the system writes the IDs of the respective object at runtime (in the example, the contact keys). This ensures that the score is not calculated for all contacts at runtime but only for the contact in focus. For the contact keys valid in the current session of a user, the system creates a temporary key that is submitted to the procedure as a parameter. For contacts, SAP delivers the table `CUAND_JS_CONTACT`, which you can use. The table is to be specified when executing the Customizing for Segmentation, see section [Executing Customizing Activities in Customizing for Segmentation \[page 25\]](#).




Copy the above mentioned information models. To copy the information models of the example score, see section [Copying SAP HANA Information Models](#) in section [Extending the Data Source and Business Objects \[page 16\]](#). Adapt your copies of the information models according to the requirements of the score that you want to create and ensure that the interdependencies between the information models are kept. Do not change the input and output parameters of the information models.

Caution

Make sure that all information models that you create are cross-client to ensure proper data flow.



Data Flow

For detailed information about one of the above mentioned SAP HANA information models, use the [Auto Documentation](#) function in SAP HANA Studio. This function provides a PDF document for each model, including all fields of the model, and a list of models that are referenced.

For more information about SAP HANA Studio, see the SAP Help Portal at http://help.sap.com/hana_platform 
 [Developer Information](#)  [SAP HANA Developer Guide](#) .

2.3.1.2 Executing Customizing Activities for Segmentation

Besides pure segmentation settings, Segmentation provides you with features for performance improvements. In addition, you can use its maintenance views as data provider with known field names.

To integrate scope and improve performance while displaying scores in profiles, and, in addition, to use scores in segmentation itself, you must execute the following segmentation Customizing activities (you find it within the Implementation Guide (IMG) for *SAP Hybris Marketing* (transaction *SPRO*) under **Segmentation**  **Segmentation** .

For more information about the required steps, see the corresponding documentation for each Customizing activity.

- **Define Aliases for SAP HANA Data Sources**

In this activity, you define a shortcut to your SAP HANA information model. You can use this shortcut in all places in the AS ABAP system where you have to reference the data source. Depending on the template graphical information model that you have copied, you enter path and name of your copy of the graphical model in field *Data Source Location*.

For the **Age Score** example, the data source alias is `CA_CE_CONTACT_AGE` with data source location `sap.hana-app.cuan.contact.score/CA_CE_CONTACT_Age_SCORE_EXT` (calculation view).

- **Define Segmentation Objects**

In this activity, you create a segmentation object to determine the selection of business data for your score. For the **Age Score** example, the segmentation object `SAP_CONTACT_EGM_CONSUMER_11SP4` is delivered by SAP. The example age score is assigned to this segmentation object.

You can either use the same segmentation object or create your own one. If you want to create your own segmentation object, ensure that this object contains the following entries:

- *Consumer Class*: `CL_CUAN_CE_CONTACT_SEGMENT`
- *Consumer Selection Class*: `CL_CUAN_CE_CONTACT_SEGMENTATN`

Background Information:

A segmentation object bundles different information models (such as master data of contacts with their transaction data or with scores). The segmentation object must contain minimum one master data information model for the following reasons:

- To be able to assign the score to one or more contacts later on in the profile or in segmentation.
- To be able to restrict the use of the score to certain attributes of the contact such as country, region, or function.

The consumer selection class implements the access to join set tables in the SAP HANA database. The join set tables contain the ID of the currently used objects (such as contacts or accounts) for a score during runtime. The tables are joined in the SAP HANA procedures (see section [Creating SAP HANA Information Models in the SAP HANA Studio \[page 23\]](#)). The tables reduce the number of scores to be calculated which increases the performance significantly.

The procedure, in which the system calculates the score (in the example score, calculation view `app.cuan.contact.score.internal/PR_CE_CONTACT_AGE`) uses a join set table. This join set table is to be propagated to the segmentation object defined in the given Customizing activity, even if you use a table delivered by SAP. This propagation is implemented using segmentation consumer selection class `CL_CUAN_CE_CONTACT_SEGMENTATN` if the join set tables delivered by SAP are used in your procedure (in the example score, calculation view `app.cuan.contact.score.internal/PR_CE_CONTACT_AGE`). that you must create in transaction `SE80`

In transaction `SE80`, see class `CL_CUAN_CE_CONTACT_SEGMENTATN` for your reference. Enter the join set table name for your score in the `CONSTRUCTOR` method of your class. To get the correct table name, refer to the corresponding SAP HANA procedure.

For more information, see the Customizing activity documentation.

- **Assign SAP HANA Data Sources to Segmentation Objects**

In this activity, you assign the required SAP HANA information models to your segmentation object using the aliases that you have defined before. To do so, you create a new entry and select your segmentation object (for the **Age Score** example, this is the `SAP_CONTACT_EGM_CONSUMER_11SP4` segmentation object) and assign the alias that you have created in the first Customizing activity of this section (see above).

Besides the score information models, you must assign the population data source alias and other data sources containing further attributes to the segmentation object.

For selecting the attributes themselves and assigning them to attribute groups, see section [Providing a Custom Data Source for Segmentation \[page 80\]](#).

Note

For the creation of a score you only need the score value attribute and can hide the rank attribute.

- **Define Segmentation Object Key Fields**

In this activity, define a key field which is needed to be able to join all the information models belonging to the segmentation object.

For the **Age Score** example, this is key field `SAP_CONTACT_KEY` entered for segmentation object `SAP_CONTACT_EGM_CONSUMER_11SP4`.

- **Assign Segmentation Attributes to Object Key Fields**

In this activity, you assign the attributes (characteristics and key figures) originating from the SAP HANA information models to the segmentation object key fields that you have defined in the previous step. Search for the entry that contains your segmentation object, your data source alias and your segmentation object key field.

- If you want to use the score in segmentation, you assign an attribute such as `CONTACT_KEY` to your entry.
- If you do not want to use your score in segmentation but in a profile only, keep the attribute field empty to avoid the display of the score in segmentation.

For the **Age Score** example, the entry consists of the following parts:

- **Segmentation Object:** `SAP_CONTACT_EGM_CONSUMER_11SP4`
- **Data Source Alias:** `CA_CE_CONTACT_AGE`
- **Segmentation Object Key Field:** `SAP_CONTACT_KEY`
- **Attribute Name:** No entry

- **Define Segmentation Profiles**

In this activity, you create a segmentation profile which is mandatory for minimum one of the following purposes:

- You want to use your score in segmentation
- You want to use your score in profiles, and want to restrict the preview in the predictive model to a certain number of contacts in a target group. This target group must have been created with a segmentation profile to which the segmentation model defined or used above, belongs.

Note

You can skip this step if you have used the `SAP_CONTACT_EGM_CONSUMER_11SP4` segmentation object delivered by SAP because in this case a suitable segmentation profile already exists in the system,

For the **Age Score** example, the entry consists of the following parts:

- **Segmentation Profile:** `SAP_CE_CONSUME_11SP4`

- *Segmentation Object:* SAP_CONTACT_EGM_CONSUMER_11SP4
- *Population Data Source Alias:* SAP_CE_CONTACT_INTERACTIONS
- *Description of Segmentation Profile:* All Consumers (SCI SP04)

i Note

The example contains not only pure master data but also the contact interactions due to performance reasons (see *Population Data Source Alias:* SAP_CE_CONTACT_INTERACTIONS)

2.3.1.3 Executing Customizing Activities in Customizing for Predictive Scenarios

In *Predictive Model Management*, every score corresponds to a predictive scenario. For this reason, you have to execute a few Customizing activities for predictive scenarios which you find in Customizing for *SAP Hybris Marketing* (transaction SPRO) under *Predictive Scenarios*.

Currently a predictive scenario is offered in two types:

- Predictive Model
- Heuristic Model (from Predictive Studio)

Also you can omit the activity for defining a predictive data source due to a missing training step.

For more information about the required steps, see the corresponding documentation for each Customizing activity.

• **Define Implementation Method**

In this activity, you define the implementation method which is the algorithm for calculating the score. You can enter any name as implementation method and method description. In the *Model Execution* field, choose the path to the SAP HANA procedure that you have created in section [Creating SAP HANA Information Models in the SAP HANA Studio \[page 23\]](#) from the value help.

For your reference, the path to the example procedure for the **Age Score** example is `sap.hana-app.cuan.contact.score.internal/PR_CE_CONTACT_AGE`.

You do not have to define any model parameters for heuristic scores.

• **Define Attributes for Applicable Scope**

In this activity, you define the applicable scope for your score. The scope defines the validity for a specific set of attributes (for example, the specification of the countries which the score will cover is to be defined later on in the heuristic model).

Heuristic scores are mainly valid for all objects without any restrictions. However, you must use your segmentation object (either object SAP_CONTACT_EGM_CONSUMER_11SP4 or the one that you have created in the previous section, [Executing Customizing Activities in Customizing for Segmentation \[page 25\]](#)) to receive the link to the segmentation consumer selection class containing the join set table for performance improvements as described in the segmentation Customizing section. In addition, you must specify a data source alias. You can use data source alias SAP_CONTACT_INTERACTIONS or any data source alias that contains the attribute that you want to use for restriction purposes. In the field *Attribute Name*, you must enter an attribute that is included in the data source alias.

For a correct assignment, see the value help of the *Segmentation Object* and *Data Source Alias* fields and search for your segmentation object and a suitable data source alias.

For your reference, the applicable scope of the **Age Score** example is `SAP_CE_TI_CONSUMER_SCOPE`. This scope contains the segmentation model `SAP_CONTACT_EGM_CONSUMER_11SP4` and the data source alias `SAP_CONTACT_INTERACTIONS` (an alias referring to a view that contains most master data attributes of a contact) as well as the attribute `COUNTRY` (as the example score is valid only for specific countries).

- **Define Application Anchors**

In this activity, you define application anchors for your score. In the context of score creation, application anchors are used by the user interface objects when calling the back end to retrieve the score values. Thus, the application anchors ensure the display of your score within the required profile on the user interface. The names for the application anchors must correspond to names for the applications that have been defined during the user interface setup. If you use the standard user interface and did not copy and adapt it, the application anchors are the following:

- `CUAN_CUSTOMER` for the *Corporate Account* details.
- `CUAN_INTERACTION_CONTACT` for the *Contact* profile.
- `CUAN_CONSUMER` for the *Consumer* profile.
- `CUAN_PROSPECT` for the *Suspect* profile.

If you use self-defined profiles, you must enter the anchor name that is part of the user interface request for calling the back end to retrieve the score values.

- **Define Predictive Scenarios**

In this activity, you define the actual predictive scenario, that is your score, by entering and assigning the following parts that you have defined in the previous steps:

- Entering any scenario description
- Choosing the scenario type *Heuristic Model*.
- Entering the path to the graphical information model for your score that you have created in section [Creating SAP HANA Information Models in the SAP HANA Studio \[page 23\]](#).
- Assigning the implementation method to the score
- Assigning the applicable scope to the score
- Assigning the application anchor to the score

For the **Age Score** example, the following parameters are provided (see below). You must enter your equivalent of the indicated parameters.

- *Predictive Scenario*: `CONTACT_AGE_SCORE` with the *SAP HANA View* `sap.hana-app.cuan.contact.score/CA_CE_CONTACT_AGE_SCORE_EXT`
- *Implementation Method*: `CONTACT_AGE_SCORE`
- *Applicable Scope*: `SAP_CE_TI_CONSUMER_SCOPE`
- *Application Anchor*: `CUAN_CONSUMER` (as the example score is to appear on the *Consumer* profile)

2.3.1.4 Maintaining the Heuristic Model on the User Interface for Predictive Models

To complete the creation of your heuristic score and make it available in your profiles, you have to create a heuristic model that corresponds to your predictive scenario in the *Predictive Studio* app on the user interface. All models have to be created manually.






Usually, you do not define a target group for heuristic models as the system calculates on the basis of all data in the system. You can create a target group to limit the data basis. If so, the target group must fit exactly to the referred object such as contact or account.

Note

On the *Predictive Studio* UI, the term *Predictive Models* includes both heuristic and predictive models.

To create a heuristic model, proceed as follows:

1. Choose the *Predictive Studio* workset on the user interface of *SAP Hybris Marketing*.
2. Choose the *Create* pushbutton.
3. In the *Key Information* section, select your predictive scenario from the dropdown menu of the *Predictive Scenario* field.
4. Choose the *In Preparation* pushbutton at the bottom of the screen.
5. You can add a scope to restrict the validity of the heuristic score to certain attribute values of the attributes specified in Customizing. To do so, choose the *Add Scope* pushbutton in the *Application Scope* section. Select the attribute, select the required attribute values, and confirm your entries.
6. Choose the *Add Model Fit* pushbutton in the *Model Fits* section. In the appearing dialog box, enter a self-defined name for the model fit. The model fit details appear.
7. Go back to the predictive model details.
8. Select the *Best Fit* checkbox in the *Model Fits* section. The *Publish* pushbutton appears at the bottom of the screen.
9. Once you have chosen, the *Publish* pushbutton, your score is available in the specified profile.

For general information about how to create a heuristic model, see the corresponding application help for *SAP Hybris Marketing* under <http://help.sap.com/mkt>  *Application Help*  *SAP Hybris Marketing Applications*  *Contacts and Profiles*  *Predictive Studio* .

http://help.sap.com/saphelp_mkt118/helpdata/en/21/3f155201049a33e10000000a44538d/content.htm?frameset .

2.4 Setting-Up Predictive Scenarios

In *SAP Hybris Marketing*, you can use statistical methods to detect patterns and trends in historical data, and use the information to predict customer behavior. To enable a specific predictive use case, you define an according predictive scenario. The predictive scenario allows you to combine the predictive use case, such as the prediction of buying propensity, the predictive data source, such as the master and transactional data of CRM business partners, and the statistical methods to be used for the predictive calculation, such as the logistic regression based on the SAP HANA Predictive Analysis Library (PAL).

To define a predictive scenario and to integrate it with SAP Hybris Marketing, you perform the following main tasks:

- You create SAP HANA calculation views and SQLscript procedures in which you refer to your predictive data source and the statistical methods to be used for the predictive calculation.
- Based on the calculation views and SQLscript procedures, you define the predictive scenario, and the details of the integration with an application, such as Segmentation. You define the predictive scenario in Customizing for SAP Hybris Marketing.

Once you have defined your predictive scenario, a business analyst can use the *Predictive Studio* app in *SAP Hybris Marketing* to create, validate, and train predictive models. Note that the PFCG role `SAP_CEI_PBA` is

required to access the workcenter. Finally, the business analyst publishes predictive models. As a result, the predictive models are available in the application, and can be used for the predictive use case, such *Buying Propensity* in Segmentation.

2.4.1 Providing SAP HANA Information Models

Use

Use SAP HANA Studio to provide SAP HANA information models and SQLScript procedures for a predictive scenario.

Note

The performance of the application depends on the performance of the SAP HANA information models and SQLscript procedures you provide. Familiarize yourself with the modeling guidelines to ensure that your information models and procedures are as performant as possible.

For more information about the modeling, see the *SAP HANA Modeling Guide*, and the *SAP HANA Developer Guide* at http://help.sap.com/hana_platform. In addition, consider the *SAP HANA Performance Analysis Guide* for information about how to identify and resolve performance issues of your SAP HANA database.




Note

Make sure you use the correct format when you indicate the path and the name of SAP HANA information models, or SQLscript procedures:

- Lower case for the path, for example, `sap.hana-app.cuan.cpred.datafoundation`
- Upper case for the name of the model, or procedure, for example, `CA_CPRED_CRM_DS_PROD_AFFIN`
- Separate the path from the name by a slash `</>`.

Proceed as follows:

- Create a calculation view to define the predictive data source.
- Create SQLscript procedures for the validation and training of predictive models.
- Create calculation views, and SQLscript procedures for the execution of the predictive model.

For additional information about how to copy standard SAP HANA information models delivered by SAP, see section  *Including Customer Specific Fields in Relationship Analysis*  *Providing SAP HANA Information Models*  in this guide. For information about the standard models delivered to support, for example, the buying propensity predictive scenario, see *Business Content* at <http://help.sap.com/mkt>.

Note

Using A Join Set ID

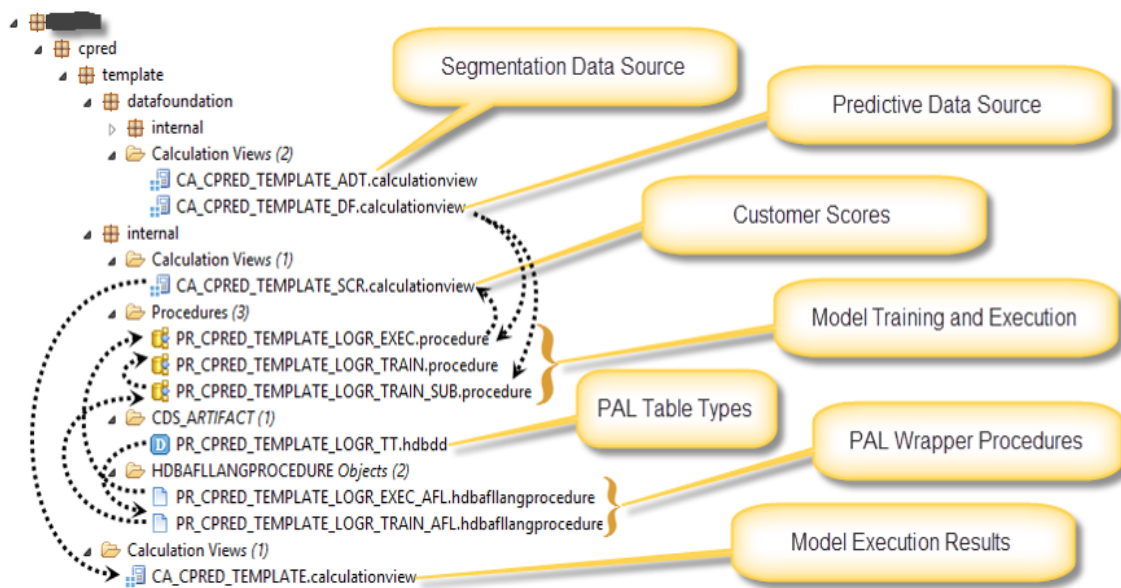
Predictive Analytics is typically based on high amounts of data. In contrast to a classic relational data access, the data is processed on large parts of an even larger complete data set. To minimize runtime and/or memory overhead, the data should be strictly limited to the amount that is needed for the current predictive analysis.

For this purpose, the concept of the join set ID has been introduced. The join set ID represents a target group as a training set for a predictive model. With this ID, instead of creating database joins with the complete base of customers, joins are created using the target group that typically consists of a smaller data set.

The backend functionality of SAP Predictive Analytics provides a framework to use the join set ID. The join set ID references a training set to temporarily create a subset of the data that is used during the process. The join set ID is then passed on to the process as a parameter.

For a summary of the standard SAP HANA information models and SQLscript procedures that are used by the predictive scenario Buying Propensity in SAP Hybris Marketing, see Business Content on the help portal at help.sap.com/mkt.

The following picture is an example of the views and package structure that are created in the following steps:



Predictive Data Source

Create a script-based calculation view to address your data source for the predictive scenario:

- Provide a name, and a description for the calculation view.
- Indicate the package where you want to locate the calculation view in the hierarchy of SAP HANA information models. For an example, see the structure of the standard package under `sap.hana-app.cuan.cpred`.

In the script section of the calculation view, refer to all fields of your data source that are relevant for the predictor variables and the target variable. For an example, see the standard calculation view under `sap.hana-app.cuan.cpred.datafoundation`.

For a summary of the standard SAP HANA information models and SQLscript procedures that are used by the predictive scenario *Buying Propensity* in SAP Hybris Marketing, see *Business Content* on the help portal at <http://help.sap.com/mkt>.

Define the following input parameters:

- Indicate the data source for the training set.
The training set provides the data describing the actual customer behavior that can be used to train the predictive model. It typically corresponds to a target group defined in segmentation, using the view you defined before.

- Target variable (Topic not covered in this guide. See content located under `sap.hana-app.cuan.cpred.datafoundation`.)
- Predictor variables
If relevant, define data parameters for (dynamic) predictor variables that are specified by the user and filled during run time.
- Target object (Topic not covered in this guide. See content located under `sap.hana-app.cuan.cpred.datafoundation`.)

Define the output parameters, such as the customer ID, and the target variable as attributes, or as measures. For measures, define the aggregation type [sum](#).

Training of the Predictive Model

Create the following SQLScript procedures to support the cross-validation and training of the predictive models:

Cross-Validation and Training (Sub)

Create an SQLScript procedure for the cross-validation and the training of the predictive models. The procedure is designed to refer to your predictive data source, and to process the cross-validation and model training methods. Proceed as follows:

- Provide a name, a description, and indicate the package where you want to locate the SQLScript procedure. For an easy reference during the setup of the predictive scenario, add the suffix `_LOGR_TRAIN_SUB` to the name.
- For [Schema](#), select `_SYS_BIC`
- For [Default Schema](#), select `SAP_CUAN`
- For [Run With](#), select *Invoker's Rights*
- For [Access Mode](#), select *Read Write*
- For [Language](#), select *SQL script*

Use the script of the standard SQLScript procedure for the cross-validation and the model training as a template. See the standard package hierarchy under `sap.hana-app.cuan.cpred.internal/PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_SUB`.

Edit the script as follows:

- Adapt the list of fields according to your predictive data source.
- Specify the calculation view you have created to address your predictive data source (providing the training set).
- Adapt the call to the Predictive Analysis Library (PAL) to the procedure you will create later (see the section **Calling the Predictive Analysis Library (PAL)** below).

Cross-Validation and Training (Generic)

This SQLScript procedure serves as a generic shell for the SQLScript procedure you have created before. Proceed as follows:

- Provide a name adding the suffix `_LOGR_TRAIN`. Provide a description, and indicate the package.
- For [Schema](#), select `_SYS_BIC`
- For [Default Schema](#), select `SAP_CUAN`
- For [Run With](#), select *Invoker's Rights*
- For [Access Mode](#), select *Read Write*
- For [Language](#), select *SQL script*

Use the script of the standard SQLScript procedure as a template. See the standard package hierarchy under `sap.hana-app.cuan.cpred.internal/PR_CPRED_PROD_AFFINITY_LOGR_TRAIN`.

In the script, specify the SQLscript procedure you have created before adding the suffix `_LOGR_TRAIN_SUB` to the name.

Calling the Predictive Analysis Library (PAL)

The PAL must be called using the Analytic Function Library (AFL) of SAP HANA. This is done in the SAP HANA Development perspective, creating a "hdbafllangprocedure" object (in contrast to the procedure objects that you created before in the Modeler perspective) that works as a wrapper for the library function.

Note

See the standard package hierarchy under HANA Predictive Analysis Library (PAL) Reference guide for a complete overview of the SAP HANA PAL.

As a precondition, the table types used in the procedure's interface have to be created. For this, proceed as follows:

- Switch to the SAP HANA Development perspective.
- In the navigation bar, select the "Repositories" tab.
- Right-click on the package where you want the file to be created (normally the same package where the procedures are located) and select New -> Other -> SAP HANA -> Database Development -> DDL Source File.
- Click "Next" and provide a name, then click "finish". The name should correspond to the names that you chose for your procedures, and end with suffix `_TT` (e.g. `PR_CPRED_PROD_AFFINITY_LOGR_TT`).
- In the created file, change the schema in the line beginning with "@Schema" to 'SAP_CUAN_APPL'.
- Provide a definition for the table type used in your training procedure (see above).
- Provide the definition of the control, model and pmml tables (see the delivered standard `sap.hana-app.cuan.cpred.internal/:PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_L.TT.hdbdd`).
- Save and activate (Activate SAP HANA Development Object).

Create an hdbafllangprocedure for the PAL call as follows:

- Right-click the package in which you want the file to be created (normally the same package where the procedures are located) and select New File.
- Provide a name that corresponds to your training procedures (see above) and extension `.hdbafllangprocedure` (for example, `PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_AFL.hdbafllangprocedure`).
- For an example, see the delivered `sap.hana-app.cuan.cpred.internal:PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_AFL.hdbafllangprocedure`. You need to adjust the packages and names of the types involved to the environment you are currently creating.
- Save and activate (Activate SAP HANA Development Object). In the `TRAIN_SUB` procedure you created earlier, replace the AFL procedure call with the procedure you just created.

Weights for the Predictor Variables

Create an SQLScript procedure for the calculation of the predictor variables weights as follows:

- Provide a name adding the suffix `_LOGR_TRAIN_WEIGHTS`. Provide a description, and indicate the package.
- For *Schema*, select `_SYS_BIC`
- Do not specify a *Default Schema* (leave empty)
- For *Run With*, select *Invoker's Rights*

- For *Access Mode*, select *Read Only*
- For *Language*, select *SQL script*

Use the script of the standard SQLScript procedure for the weights calculation as a template. See the standard package hierarchy under `sap.hana-app.cuan.cpred.internal/PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_WEIGHTS`.

In the script, adapt the list of fields according to your predictive data source.

Execution of the Predictive Model

For the execution of the predictive models, you create a calculation view, a SQL script procedure and an AFL language procedure, which are called by the calculation view.

Model Execution

Create an SQLScript procedure for the model execution as follows:

- Provide a name adding the suffix `_LOGR_EXEC`. Provide a description, and indicate the package.
- For *Schema*, select `_SYS_BIC`
- Do not specify a *Default Schema* (leave empty)
- For *Run With*, select *Invoker's Rights*
- For *Access Mode*, select *Read Only*
- For *Language*, select *SQL script*

Use the script of the standard SQLScript procedure for the model execution as a template. See the standard package hierarchy under `sap.hana-app.cuan.cpred.internal/PR_CPRED_PROD_AFFINITY_LOGR_EXEC2`.

In the script, adapt the list of fields according to your predictive data source.

Customer Scores

Create a script-based calculation view. The calculation view is designed to hold the customer scoring as calculated by SQLScript procedures you call from the calculation view. Proceed as follows:

- Provide a name adding the suffix `_SCR`. Provide a description, and indicate the package.
- For *Default Schema*, select `SAP_CUAN`
- For *Run With*, select *Invoker's Right (User_SYS_REPO)*

Use the script of the standard calculation view for the customer scores as a template. See the standard package hierarchy under `sap.hana-app.cuan.cpred.internal/CA_CPRED_PROD_AFFINITY_SCR`.

In the script, adapt the section *Execute Model* as follows:

- Specify the statistical method (in the script, it is called *predictive process*).
- Specify the SQLScript procedure you have created for the model execution (adding the suffix `_LOGR_EXEC` to the name).

Calling the PAL for Model Execution

As for the previous case (Training of the Predictive Model), the PAL must be called using the Analytic Function Library (AFL) of SAP HANA.

As a precondition, the table types used in the procedure's interface have to be created. For this, proceed as follows:

- Switch to the SAP HANA Development perspective.
- In the navigation bar, select the Repositories tab.

- Locate the DDL Source file you created earlier for model training (for example, PR_CPRED_PROD_AFFINITY_LOGR_TT).
- Provide a definition for the table type used in your execution procedure (see above) in addition to the table types that you created earlier.
- Save and activate (Activate SAP HANA Development Object).

Create a hdbafllangprocedure for the PAL call as follows:

- Right-click on the package where you want the file to be created (normally the same package where the procedures are located) and select New -> File.
- Provide a name that corresponds to your execution procedure (see above) and extension .hdbafllangprocedure (e.g. PR_CPRED_PROD_AFFINITY_LOGR_EXEC_AFL.hdbafllangprocedure).
- For an example, see the delivered sap.hana-app.cuan.cpred.internal:PR_CPRED_PROD_AFFINITY_LOGR_EXEC_AFL.hdbafllangprocedure. You need to adjust the packages and names of the types involved to the environment you are currently creating.
- Save and activate (Activate SAP HANA Development Object).
In the EXEC procedure you created earlier, replace the AFL procedure call with the procedure you just created.

Model Execution Results

Create a graphical calculation view. The calculation view serves as a generic shell to provide the results of the model execution to the consuming application. Proceed as follows:

- Provide a name, a description, and indicate the package
- For *View Type*, select *Graphical*
- For *Default Schema*, select *SAP_CUAN*
- For *Run With*, select *Definer's Right (User_SYS_REPO)*

Specify the calculation view you have created to hold the customer scoring (adding the suffix **_SCR** to the name). For an example, see standard package hierarchy under sap.hana-app.cuan.cpred/CA_CPRED_PROD_AFFINITY.

2.4.2 Customizing a Predictive Scenario

You define a predictive scenario in Customizing for SAP Hybris Marketing. Besides the definition of the predictive scenario you need to define the integration of the predictive scenario with an application, such as Segmentation.

i Note

For the details, see the documentation as provided for the Customizing in the system. Also, use the F1 help, which provides additional context sensitive help.

Defining a Predictive Scenario

Use the following Customizing activities in ► [SAP Hybris Marketing](#) ► [Contacts and Profiles](#) ► [Predictive Scenarios](#) ►:

- [Define Number Range for Predictive Models](#)
The business analyst provides predictive models for use in the applications. Each model carries a unique number. In this activity you define the number range for the model numbering.
- [Define Predictive Data Source](#)
In this activity, you define a reference to the graphical calculation view you have created to provide the results of the model execution to the consuming application (see section [Providing SAP HANA Information Models](#)).
- [Define Implementation Method](#)
In this activity, you determine the statistical method to be used in the predictive scenario. You define the statistical methods, which are available in Customizing, in the script of the calculation view for the customer scores.
- [Define Attributes for Applicable Scope](#)
In the activity, you select the attributes that can be used to define an applicable scope (in the predictive model maintenance). The applicable scope restricts the validity of the predictive model in the context of the current scope. The more general the applicable scope of the model is defined, the more restrictive the current context or scope can be so that the model can still be applied.
You will need to provide the Datasource alias of the HANA view that represents the datafoundation of your scenario. You define this Datasource alias in ► [SAP Hybris Marketing](#) ► [Segmentation](#) ► [Define Aliases for SAP HANA Data Sources](#) ► (see next chapter [Integration with Segmentation](#)).
Example:
A model with applicable scope Country = 'United States' can be applied in the context of State = 'California' or City = 'New York'.
A model of applicable scope Country = 'United States' and State = 'California' cannot be applied in the context of City = 'New York'. It can be, however, applied in the context of City = 'Los Angeles'.
- [Define Client Applications](#)
Note that client applications only need to be assigned to predictive scenarios in this customizing activity if the client application does not define the scenario itself. This is the case, for example, for predictive scores on the contact fact sheet. In [Segmentation](#), the predictive scenario is explicitly chosen during creation of the segmentation model.
For more details on client applications, consult the application help that is available in the customizing activity.
- [Define Predictive Scenarios](#)
You determine the model parameters (and default values if applicable), the statistical method, and the applicable scope.
- [Define Settings for Display of Predictive Scenarios on User Interface](#)
For more details on settings for the display of predictive scenarios, consult the application help that is available in the customizing activity.
- [Define Attribute Hierarchy](#)
Define optional hierarchical dependencies between related attributes, such as country and region.

Integration with Segmentation

Use the following Customizing activities in ► *SAP Hybris Marketing* ► *Segmentation* ► to integrate the predictive scenario with the application:

- **Define Aliases for SAP HANA Data Sources**
In this activity, you define a reference to the graphical calculation view you have created to provide the results of the model execution to the consuming application. Also, you define a reference for your segmentation datasource.
- **Define Segmentation Preview Types**
In this activity, you define a preview type that allows to apply the predictive scenario within *Segmentation*. Define the preview type as follows:
 - Specify the *Preview Type*, if you have created one. To reuse the standard gain chart preview, enter: **SAP_KF_PRED_GAIN_CHART**.
 - For the *Preview Type Usage* choose *Key Figure*.
 - If applicable, specify the SAPUI5 *Preview Type Path* where the preview type is located. To reuse the standard gain chart preview, enter: **/sap/bc/ui5_ui5/sap/gseg_pred_prev/gseg_pred_prev**
 - If applicable, specify the SAPUI5 *Preview Name*. To reuse the standard gain chart preview, enter: **gseg_pred_prev.predictivePreview**
 - If applicable, provide a description for the preview type, such as **Gain Chart**.
- If you cannot reuse the standard segmentation object for the predictive scenario (SAP_CRM_BP_PREDICTIVE), you will need to define a custom segmentation object. For more details, see the section *Creating Predictive Scores in Account and Contact Fact Sheets*.
- **Assign SAP HANA Data Sources to Segmentation Objects**
In this activity, you assign the calculation view for the predictive results to segmentation objects, and you define details for the predictive attributes, which can be used in the application. Proceed as follows:
 - If you have defined a custom segmentation object for the predictive scenario, select the according segmentation object. To reuse the standard segmentation object for the predictive scenario, select **SAP_CRM_BP_PREDICTIVE**. To complete the assignment, select the alias you have defined for the reference to the calculation view providing the predictive results.
 - Mark the assignment you have created, and click *View data source attributes*. Enter the attributes that are relevant for the predictive scenario. Per attribute, specify the details, such as the preview type in which the attribute can be used. For an example, see the details of the standard assignment of the segmentation object **SAP_CRM_BP_PREDICTIVE**, and the data source alias **SAP_CRM_PRODUCT_AFFINITY**.
- **Assign Segmentation Attributes to Object Key Fields**
In this activity, you define the attribute serving as a key for the predictive results, such as the ID of the customer for the buying propensity prediction. Select the segmentation object, the data source alias, the segmentation object key field, and the attribute for an assignment. For an example, see the **SAP_CRM_PRODUCT_AFFINITY** assignment.

2.4.3 Defining Predictive Scenarios Based on SAP HANA Rules Framework

To set up a heuristic scenario based on the SAP HANA rules framework (HRF) you perform the following tasks:

- Create a HRF vocabulary

Note

Please note that in *Score Builder* you can only use the provided HRF vocabulary. You cannot use the application with a customized HRF vocabulary. However, you can create heuristic scores based on your own HRF vocabulary in the *Predictive Studio* application. For more information, also see [Creating Predictive Scores in Account and Contact Fact Sheets \[page 22\]](#).

- Create SAP HANA information models for the predictive scenario
- Configure the predictive scenario in Customizing

Prerequisites

Before you set up the predictive scenario, check the following:

- You have installed the SAP HANA rules framework (HRF).
For more information about how to install HRF, see the section [Optional Configuration Settings for SAP hybris Marketing > Scoring Based on HANA Rules Framework \(HRF\)](#) in the *Installation Guide* for SAP Hybris Marketing at <http://help.sap.com/mkt> [Installation and Upgrade Information](#).
- You have created SAP HANA information model(s) to provide the business data for the predictive scenario.
Note that you reference the SAP HANA information models in the HRF vocabulary, which you create in the course of setting up the predictive scenario.
For more information about the modeling, see the *SAP HANA Modeling Guide*, and the *SAP HANA Developer Guide* at http://help.sap.com/hana_platform.
- Make sure the user you require to create a HRF vocabulary in SAP HANA studio, has the following package privileges
 - REPO.READ
 - REPO.MAINTAIN_NATIVE_PACKAGES
 - REPO.EDIT_NATIVE_OBJECTS
 - REPO.ACTIVATE_NATIVE_OBJECTS

Vocabulary

To set up a predictive scenario based on HRF you need to create a HRF vocabulary. For more information about how to create a HRF vocabulary, see the *SAP HANA Rules Framework - Development & Implementation Guide*, section [Resources of SAP HANA Rules Framework > Vocabulary](#) on the *SAP Support Portal* at <http://service.sap.com/instguides> [SAP In-Memory Computing > SAP HANA Rules Framework 1.0](#).

You create a HRF vocabulary as SAP HANA content in SAP HANA studio. Proceed as follows:

Create a Package

1. In SAP HANA studio, open the *SAP HANA Development* perspective.
2. Click the *System* tab. Choose your system and create a new package in the *Content* folder. For example:
[> <system ID> > Content > tmp > <user> > vocabulary](#).

Create a Project

1. In SAP HANA studio, right-click the *Project Explorer* tab, and select *New Project ...* in the context menu.
2. Choose **SAP HANA Application Development** > **XS Project** on the *New Project* window, and choose *Next*.
3. Enter a project name, for example, **Vocabularies**. Ensure that *Share project in SAP repository* is selected. Choose *Next*.
4. Select your *Repository Workspace*, and choose *Browse...* to select the package that you have created before (see section *Create a Package*). Choose *OK*, and subsequently choose *Finish*.

Create a Vocabulary

1. In SAP HANA studio, right-click the project on the *Project Explorer* tab. Choose **New** > **File** in the context menu.
2. Enter a file name with the suffix **.hrfvocabulary**.
3. Copy and paste the source code for the HRF vocabulary into the registered editor. Save your work.

i Note

You need to include a defined output in the source code of the vocabulary (see section *Vocabulary Output*).

4. Right-click the *Project Explorer*, and choose *Refresh* in the context menu (or choose F5).

Vocabulary Output

In the HRF vocabulary you create, include the following output definition:

```
Syntax
"outputs": [{
  "name": "Output",
  "description": "<this description does not appear on UI>" "inputParams": [{
    "name": "score",
    "dataType": "<numeric data typen>",
    "size": "<size declaration of datatype>",
    "businessDataType": "Number"
  }]
}]
Foranexample,
seethefollowingoutputdefintion: Syntax"outputs": [{
  "name": "Output",
  "description": "Output" "inputParams": [{
    "name": "score",
    "dataType": "INTEGER",
    "size": "",
    "businessDataType": "Number"
  }]
}]
}]
```

Note that in the example, the size of the datatype is empty because the INTEGER does not require a size.

For a complete example of the vocabulary, see following source code:

```
Syntax
{
  "dataObjects": [{
    "name": "Customer",
    "description": "CRM Business Partners",
    "attributes": [{
      "name": "CustomerID",
      "description": "Customer ID",
      "dataType": "NVARCHAR",
      "size": "10",
      "businessDataType": "String",
      "sourceType": "Data",
      "dataMapping": {
```



```

        "column": "PARTNER"
    }
    },
    "associations": [{
        "name": "SalesOrders",
        "target": "crm_sales_orders",
        "cardinality": "OneToMany",
        "attributeMappings": [{
            "source": "CustomerID",
            "target": "SoldToParty"
        }],
    }],
    },
    "mappingInfo": {
        "schema": "_SYS_BIC",
        "name": "sap.hana-app.cuan.cpred.datafoundation.internal/
CA_HRF_CRM_BP_JS (placeholder.\\"$sip_js_id$$\" => :JOIN_SET_ID )",
        "type": "CalculationView"
    }
},
{
    "name": "crm_sales_orders",
    "description": "CRM Sales Orders",
    "attributes": [{
        "name": "SoldToParty",
        "description": "Sold-to Party",
        "dataType": "NVARCHAR",
        "size": "10",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "PARTNER"
        }
    }],
    {
        "name": "OrderID",
        "description": "Sales Order Id",
        "dataType": "NVARCHAR",
        "size": "10",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "OBJECT_ID"
        }
    },
    {
        "name": "SalesOrg",
        "description": "Sales Organization",
        "dataType": "NVARCHAR",
        "size": "15",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "SALES_ORG"
        }
    },
    {
        "name": "DistributionChannel",
        "description": "Distribution Channel",
        "dataType": "NVARCHAR",
        "size": "5",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "DIS_CHANNEL"
        }
    }
},
{

```

```

        "name": "Division",
        "description": "Division",
        "dataType": "NVARCHAR",
        "size": "5",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "Division"
        }
    },
    {
        "name": "Date",
        "description": "Document Date",
        "dataType": "DATE",
        "size": "",
        "businessDataType": "Date",
        "sourceType": "Data",
        "dataMapping": {
            "column": "POSTING_DATE"
        }
    },
    {
        "name": "Product",
        "description": "Product",
        "dataType": "NVARCHAR",
        "size": "32",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "PRODUCT"
        }
    },
    {
        "name": "ProdcutCategory",
        "description": "Prodcut Category",
        "dataType": "NVARCHAR",
        "size": "32",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "CATEGORY_GUID"
        }
    },
    {
        "name": "SalesVolume",
        "description": "Sales Volume",
        "dataType": "DECIMAL",
        "size": "15,3",
        "businessDataType": "String",
        "sourceType": "Data",
        "dataMapping": {
            "column": "NET_VALUE"
        }
    }
],
"mappingInfo": {
    "schema": "_SYS_BIC",
    "name": "sap.hana-app.cuan.cpred.datafoundation.internal/
CA_HRF_CRM_SLSO_JS (placeholder.\"$$ip_js_id$$\" => :JOIN_SET_ID )",
    "type": "CalculationView"
}
}],
"outputs": [{
    "name": "Output",
    "description": "Output"
    "inputParams": [{
        "name": "score",
        "dataType": "INTEGER",
        "size": "",
        "businessDataType": "Number"
    }
    ]
}
]

```

```

    } ]
  } ]
}

```

Activate the Vocabulary

In SAP HANA studio, right-click the vocabulary on the *Project Explorer* tab. Choose ► **Team** ► **Activate** ► in the context menu.

Complete the vocabulary activation in the back end system as follows:

1. Use *ABAP Editor* (transaction SE38) to execute the program *Activate HRF Vocabulary* (CUAN_HRF_ACTIVATE_VOCABULARY).
2. Use the input help to select the vocabulary you have activated in SAP HANA studio.
3. Choose *Execute*.

SAP HANA Information Models for the Predictive Scenario

In SAP HANA studio, create calculation views that can be used by SAP Hybris Marketing applications, such as *Segmentation* to access the HRF-based scoring results. Use the following calculation views (delivered with the demo scenario CRM loyalty score) as templates to create your custom calculation views:

- system-local.sap.hana-app.cuan.cpred.demo.loyalty.internal/CA_CPRED_LOYALTY_CRM_SCR
- system-local.sap.hana-app.cuan.cpred.demo.loyalty/CA_CPRED_LOYALTY_CRM

Proceed as follows to adapt your copy of the calculation view system-local.sap.hana-app.cuan.cpred.demo.loyalty/CA_CPRED_LOYALTY_CRM:

1. Select the *Aggregation* node. Delete the reference to calculation view system-local.sap.hana-app.cuan.cpred.demo.loyalty.internal/CA_CPRED_LOYALTY_CRM_SCR (as you need to assign your custom calculation view).
2. On the *Aggregation* node, choose *Add Objects*. Add the custom copy of the calculation view system-local.sap.hana-app.cuan.cpred.demo.loyalty.internal/CA_CPRED_LOYALTY_CRM_SCR, which you have created before.
3. In the *Details* area, select the key column (for example, *BP_ID*), and choose *Add To Output*.
4. In the *Details* area, select the source column (for example, *SCORE*), and choose *Add As Aggregated Column*.
5. Select the *Semantics* node. Adapt the *Label* of the score column as required.
6. In the *Variables/Input Parameters* settings, choose *Input Parameter Manage Mapping*. In the *Manage Mappings* dialog, choose *Auto Map By Name* (to map the input parameters of your custom calculation views).

Customizing

To complete the setup of the predictive scenario you configure the scenario in Customizing for *SAP hybris Marketing* under *Predictive Scenarios*. Proceed as follows:

1. Choose *Define Predictive Data Source*. Create an entry defining a *Data Source Alias*, and a data source description. Specify the data source as follows: <path of HRF vocabulary>::<name of HRF vocabulary>. *Key Column*: <root data object>:<key attribute name>

Example

The data source for the demo scenario CRM loyalty score is defined as follows:

Predictive Data Source: `system-local.sap.hana-app.cuan.cpred.hrf::SAP_Demo_CRM_Loyalty`

Key Column: `Customer:CustomerID`

2. Choose *Define Implementation Method*. Create an entry, and specify the following:
 - Name and description for the implementation method
 - For the *Model Execution Procedure*, enter: `system-local.sap.hana-app.cuan.cpred.reuse.internal/PR_PRED_HRF_EXEC`
 - Data source alias (see step 1.)
 - For the *Implementation Method Category*, enter: `SAP_HRF`.

Example

See the implementation method `DEMO_SCORE_CRM_LOYALTY_HRF`.

3. Choose *Define Predictive Scenarios*. Create an entry, and specify the following:
 - Name and description for the scenario
 - For the *Scenario Type*, select *Heuristic Model*.
 - Under *SAP HANA View*, specify the path, and the name of the calculation view that allows applications in SAP Hybris Marketing to access the scoring results.

Example

If you have used the template delivered with the demo scenario CRM loyalty score to create such a calculation view, then you specify the name and path of your custom copy of the following calculation view: `system-local.sap.hana-app.cuan.cpred.demo.loyalty/CA_CPRED_LOYALTY_CRM`.

See section *SAP HANA Information Models for the Predictive Scenario* for the details.

2.5 Uploading Externally Trained Model Fits

Before you read on, please consider that the model fit upload works out of the box if the following applies:

- The predictive scenario is *Consumer Buying Propensity*.
- The implementation method is *Logistic Regression*.
- The model is in *PMML (Predictive Model Markup Language)* format based on *SAP HANA Predictive Analysis Library (PAL)* or *IBM SPSS*.

To upload externally trained models in PMML format for other predictive processes, such as decision trees, or for other software, such as SAS, you need to

1. Transform the PMML model into a PAL-supported format
2. Provide the necessary SAP HANA artifacts.
3. Execute steps in Customizing for *SAP Hybris Marketing*.

Requirements for a Successful Upload

Before implementing the transformation logic for a specific predictive process, please first check the basic rules of the PMML model format for the corresponding predictive process. For example, these are the rules for logistic regression:

- Categorical predictors can only come in the data type integer or string.
- The target column that you want to predict can only be categorical (and must come in the data type integer or string).
- The data types of predictors from externally trained models must match the data types of corresponding predictors from the predictive data source in the SAP Hybris Marketing system.

1. Transform the PMML Model Fit into a PAL-Supported Format

SAP provides the interface `IF_CUAN_PM_PMML_IMPORT` for the implementation of PMML transformation logic.

As standard delivery, SAP provides two classes:

- `CL_CUAN_PM_PMML_SPSS_LOGR` for IBM SPSS Logistic Regression
- `CL_CUAN_PM_PMML_PAL_LOGR` for SAP PAL Logistic Regression

Both classes implement the interface `IF_CUAN_PM_PMML_IMPORT`. To understand how class and interface interact, you can use these classes as a reference point.

To enable PMML Import for a new predictive process or Logistic Regression model based on other software, you need to implement your own class from the interface `IF_CUAN_PM_PMML_IMPORT`.

In this interface, three methods are defined: `TRANSFORM_MODEL`, `VALIDATE_MODEL`, `GET_MODEL_INFORMATION`.

1.1 TRANSFORM_MODEL

This method transforms the externally trained PMML model it into a PMML model in PAL format.

For example, XSLT is used for transformation within the class `CL_CUAN_PM_PMML_SPSS_LOGR` for IBM SPSS Logistic Regression. However, you can use methods other than XSLT for transformation.

➔ Recommendation

You can use the output table for model configuration to give further information about the model.

For example, the classes `CL_CUAN_PM_PMML_SPSS_LOGR` and `CL_CUAN_PM_PMML_PAL_LOGR` for SAP PAL use the output table for model configuration to give the list of predictors applied in training. The additional information on the predictors is then displayed within the Predictive Studio application.

1.2 VALIDATE_MODEL

This method is optional, but recommended. The method validates the PMML model to be imported.

1.3 GET_MODEL_INFORMATION

This method renders model information.

You can use ET_MODEL_DETAILS to render general information about the model.

Example

For Logistic Regression, the name, value and coefficient of the predictor are provided in ET_VARIABLE_DETAILS. This information is then displayed within the Predictive Studio application.

2. Provide the Necessary SAP HANA Artifacts

You need to provide the following:

- A SAP HANA data source.
You can create a new one or take an existing one.
- A 'hdbdd' context file that defines an interface for calling the PAL.
For an example, see the delivered sap.hana-app.cuan.cpred.internal:PR_CPRED_PROD_AFFINITY_LOGR_EXEC_AFL.hdbafllangprocedure. You need to adjust the packages and names of the types involved to the environment you are currently creating.
- A Hdbafllang procedure that wraps the direct call to PAL.
For an example, see the delivered sap.hana-app.cuan.cpred.internal:PR_CPRED_PROD_AFFINITY_LOGR_TRAIN_AFL.hdbafllangprocedure.
- An execution procedure
You can copy this example procedure and adapt it according to your use case:sap.hana-app.cuan.cpred.ic_affinity.internal/PR_CPRED_IC_PROD_AFFIN_LOGR_EXEC).
- A graphical scenario view

Note

For more information, see the chapter [Providing SAP HANA Information Models \[page 30\]](#), especially the sections *Predictive Data Source*, *Model Execution* and *Calling the PAL for Model Execution*.

3. Execute Steps in Customizing for SAP Hybris Marketing

Carry out the following steps. For more information, see Customizing documentation.

- *Define Data Source*
Ensure the type and sequence of the predictor variables are identical to those in the data source.
- *Define Implementation Method*
Model training is not required, since you can only upload trained models.
Enter the PMML Import Class you created in the interface in step 2.It will appear in the F4 help.
- *Define a Predictive Scenario*
Assign the implementation method you defined in the predictive scenario.
Note that data source parameters cannot be defined.

2.6 Enabling Persistence for Predictive Models

For the predictive scenario *Consumer Buying Propensity*, persistence does not require configuration. To enable persistence for all predictive models from other predictive scenarios, provide the following:

- A SAP HANA view or table function that

- calculates the score.
- has a predictive scenario defined in Customizing.

Only then can its scores be used and displayed in segmentation and on fact sheets.

- A SAP HANA view or table function that serves as a scenario view and that

- determines the current version of a score.
- can call up the score from the persistence table, if it is filled.
- or, if the table is not filled, can call up a score calculated in real time from the view or table function mentioned above.

Note

As an example, you can have a look at the view `sap.hana-app.cuan.cpred.ic_affinity/CA_CPRED_IC_PROD_AFFIN` in the repository of SAP HANA Studio.

- A persistence table with the following prerequisites:
 - a field for the client in the first position
 - an include for the structure CUANS_SP_VERSION_KEY
 - a subset of fields from the view, including at least the key fields, for example, contact key and score.

You define entries in two Customizing activities:

1. *Enable Persistence*

- Choose *New Entry*.
- Define a *Persistence Alias*.
- Enter the view that calculates the score.
- Enter the table in which you want to store the persisted scores.

2. *Define Predictive Scenario*

- Define a predictive scenario for the SAP HANA scenario view you created. For more information, see the documentation for this customizing activity.
- Enter the *Persistence Alias* you defined before in the customizing activity *Enable Persistence*.

You can also choose to define how often you want to persist a score. For more information, see the documentation for the customizing activity *Define Frequencies for Persistence*.

2.7 Adding Filter Attributes for Contact Engagement

You can add filter attributes, and make them available in the *Focus* panel of the *Contact Engagement* subworkset of *Social Contact Intelligence*. Additional attributes can be used in the following areas:

- Filter value help for the selection of an attribute
- Attribute search (fuzzy type)
- Range slider (for numeric attributes)

The procedure of adding an attribute is mainly the same for the different areas. For the detailed description of the complete procedure, see the section about adding an attribute for the filter value help.

Prerequisite

You have a database user with development authorization in the SAP HANA Studio. For more information, see the guides at <http://help.sap.com/hana> ► *SAP HANA Platform* ► *Development Information* ►.

Adding Attributes to the Filter Value Help

To add a filter attribute you perform the following main tasks:

- Extend the relevant include structure in the ABAP backend.
- Copy and adapt the relevant SAP HANA information models.
- Configure the additional attributes in Customizing.

Extending the Include Structure

To extend the relevant include structure, proceed as follows:

1. In the backend system, start the *ABAP Dictionary* (transaction SE11). Select *Data type*. Depending on your persistence requirements enter one of the following include structures:
 - INCL_EEW_CUAN_CE_IC_ROOT if you want to persist the attribute in the interaction contact.
 - INCL_EEW_CUAN_CE_IC_ROOT_TRANS if you do not want to persist the attribute, for example, because you use a different data source.

Choose *Change*.

2. In the *Change Structure* dialog, choose *Append Structure ...*. Enter a name for the append structure.
3. In the *Change Structure* dialog of the newly created append structure, enter the attribute(s) you want to add. Specify the attribute properties as required.
If you need a field for free text, or for a description of the attribute, enter an additional attribute using the naming convention: <name of attribute>_FT.
4. Save and activate the append structure.

Adapting SAP HANA Information Models

To provide the required data for the additional attributes you adapt the relevant SAP HANA information models as follows:

1. Use the [Mass Copy](#) function of SAP HANA Studio to create custom versions of the following models:

- sap.hana-app.cuan.contact/AT_CE_IC_FILTER
- sap.hana-app.cuan.contact.internal/AT_CONTACT_INTERACTION
- sap.hana-app.cuan.contact.internal/CA_CE_IC_FILTER
- sap.hana-app.cuan.contact.internal/CA_CE_CONTACT_INTERACTION

For more information about how to use the [Mass Copy](#) function, see section [Extending the Data Source and Business Objects](#) in this guide.

2. In the custom version of the above named SAP HANA information models, enhance the output adding the new filter attributes. Save and activate the models.
3. Clear the system buffers in Customizing for [SAP hybris Marketing](#) ► [General Settings](#) ► [Extensibility](#) ► [Clear BOPF Metadata from Buffer](#) ►. For more information, see section [Extending the Data Source and Business Objects](#) in this guide.

Configuring the Additional Filter Attributes

To make the additional attributes available in the value help, define the following settings for each additional attribute in Customizing for [SAP hybris Marketing](#) ► [Social Contact Intelligence](#) ► [Contact Engagement](#) ► [Define Settings for Contact Engagement Filters](#) ►:

Table 2:

Attribute	Text	Type	Label	Text Table	Description
Enter the name of the additional filter attribute	Enter the optional description, or free text field (note the naming convention <name of attribute>_FT)	Select the filter type Drop down (in order that the attribute appears in the value help)	Enter the name of the attribute as to be displayed in filter value help of the Focus panel in the Contact Engagement UI.	Enter the optional text table that belongs to the check table	Enter the description as used in the text table

Adding Filter Attributes for the Fuzzy Search

The procedure of adding filter attributes for the fuzzy search in the [Focus](#) panel of [Contact Engagement](#) is very similar to the procedure of adding attributes for the filter value help. Proceed as follows:

1. Extend the relevant include structure as described in section [Adding Attributes to the Filter Value Help](#).
2. Adapt the SAP HANA information models as described in section [Adding Attributes to the Filter Value Help](#).
3. Clear the BOPF buffer as described in section [Adding Attributes to the Filter Value Help](#).

4. Configure the additional attributes for the fuzzy search defining the following settings for each additional attribute in Customizing for *SAP hybris Marketing* ► *Social Contact Intelligence* ► *Contact Engagement* ► *Define Settings for Contact Engagement Filters* ►:

Table 3:

Attribute	Type	Label
Enter the name of the additional filter attribute	Select the filter type Search (in order that the attribute is available in the fuzzy search)	Enter the name of the attribute as to be displayed in the fuzzy search of the Focus panel in the Contact Engagement UI.

5. Create a full text index for the attribute to enable the search including blanks.

Example

```
CREATE FULLTEXT INDEX <INDEXNAME> ON <SCHEMANAME>.<TABLE_NAME> ('<TABLE_COLUMN>')  
ASYNC FUZZY SEARCH INDEX ON FAST PREPROCESS ON
```

Adding Numeric Attributes for the Range Slider

The procedure of adding numeric attributes for the range slider in the *Focus* panel of *Contact Engagement* is similar to the procedure of adding attributes for the filter value help. Proceed as follows:

1. Extend (as described in section *Adding Attributes to the Filter Value Help*) the relevant include structures regarding your persistence requirements, and the additional attributes you need for the minimum and the maximum values of the numeric attribute. Proceed as follows:
 1. For the numeric attribute, extend one of the following include structures depending on your persistence requirements:
 - `INCL_EEW_CUAN_CE_IC_ROOT` if you want to persist the numeric attribute
 - `INCL_EEW_CUAN_CE_IC_ROOT_TRANS` if you do not want to persist the numeric attribute
 2. Extend the include structure `INCL_EEW_CUAN_CE_IC_ROOT_TRANS` to add attributes for the minimum and maximum value using the following naming convention:
 - `<name of numeric attribute>_MAX` for the maximum value attribute
 - `<name of numeric attribute>_MIN` for the minimum value attribute
2. Adapt the SAP HANA information models as described in section *Adding Attributes to the Filter Value Help*. However, do not add the attributes for the minimum and maximum value to the output.
3. Clear the BOPF buffer as described in section *Adding Attributes to the Filter Value Help*.

4. Configure the additional attributes for the range slider defining the following settings for each additional attribute in Customizing for [SAP hybris Marketing](#) ► [Social Contact Intelligence](#) ► [Contact Engagement](#) ► [Define Settings for Contact Engagement Filters](#) ►:

Table 4:

Attribute	Type	Label
Enter the name of the additional numeric filter attribute	Select the filter type Range Slider (in order that the numeric attribute is available for the range slider)	Enter the name of the attribute as to be displayed in the range slider of the Focus panel in the Contact Engagement UI.

2.8 Manage Images

Manage images that are displayed in SAP Hybris Marketing .

This app allows you to upload different images from a central place for use in SAP Hybris Marketing . Earlier, you had to upload images for different object types in several customizing activities of SAP graphical user interface. You can upload images both for customizing as well as transactional entries.

The images belong to object types. These object types are displayed in several user interfaces. With the Manage Images app, you can upload or change an image from a central place so that the change affects all user interfaces where the image is used.

Some images are delivered with SAP Hybris Marketing . These images can be seen automatically in the respective sections (for example in ► [Planning](#) ► [Budget Plans](#) ►; the path can differ according to the roles assigned). These images are displayed under the Image column in the Manage Images app. The images delivered with SAP Hybris Marketing have names that are prefixed with SAP_<image name>. You can replace these images with images of your own choice.

Use this app to do the following:

- Upload an image: You can upload an image of your choice. This image is then used in the respective section. When you upload an image, the icon in the respective section is replaced with an image. For example, if you upload a [Media Type](#) image, the image will be displayed in ► [Planning](#) ► [Budget Plans](#) ►.

Note

The path can differ according to the roles assigned.

- Upload images in bulk: You can upload a set of images at a stretch using this option. To know how to upload images in bulk, see [Uploading Images in Bulk \[page 51\]](#).
- Download CSV template: You can download the CSV template for entering information about images uploaded in bulk.
- Download Zip File: You can download the zip file that contains images and updated CSV file after selecting a type.
- Download an image: You can download an image from the list of images available.

- Delete an image: You can delete an existing image from the list of images available in the app. As a result, you will not see this image in the list of images. When you delete an image, the image in the respective section is replaced with a default icon.
- Search for an image: You can search for an image using the Search option. Remember, your search is restricted by the type of images that you select from the drop-down. If you select *All Types* from the drop-down, your search will span over all types.
- View the number of records under a type: You can view the number of records under a type right next to the type drop-down.
- Visit the profile page of Contact and Corporate Account types: You can navigate to the profile page of each Contact or Corporate Account type by clicking the respective hyperlink in the *ID* column. To navigate to the factsheet, you will need the role SAP_CEL_TG_INI if you are using SAP Hybris Marketing.
- Filter images by types: You can filter images according to types. Once you filter the images by type, you will see only the particular type of images listed in the table.

2.8.1 Uploading Images in Bulk

To upload images in bulk, you need to first download the CSV template, enter information according to the instructions inside, and then upload the images together with the template as a zipped file.

Procedure

1. Choose [Download](#) > [Download CSV Template](#) to download the CSV template.
2. Strictly follow the instructions mentioned inside the CSV template.
3. Enter information related to the images that you wish to upload in the CSV template. Remember, the template is downloaded and automatically saved as `Template_Images.csv`.
4. Zip the CSV template and all the images that you wish to upload.
5. Click [Upload](#) to upload the zipped file.

Alternatively, choose [Download](#) > [Download Zip File](#) after selecting a type to download a zip file that contains all the images of a certain type such as Media Type and the updated CSV file. You can then upload the zip file.

Note

The [Download Zip File](#) feature is not supported for types - contact, corporate account, and user.

3 SAP Hybris Marketing Insight

3.1 Lead Management

3.1.1 Extending the Business Partner OData Service in SAP Hybris Marketing

Within the OData service for importing business partner information `CUAN_BUSINESS_PARTNER_IMP_SRV`, the two entities *Person*, and *Company* can be extended. It is not possible to extend the entities *Relationship*, and *ImportHeader*. You can extend entities as follows:

- Fields that should only be available in an SAP Hybris Marketing interaction contact have to be added to the structure `INCL_EEW_CUAN_CE_IC_ROOT` with `SE11`.
- Fields that should be available in an SAP Hybris Marketing interaction contact, and interaction have to be added to the structure `INCL_EEW_CUAN_CE_IA_IC` with `SE11`.
- Besides the extension include you can use one, or multiple of the pre-defined fields which are provided by standard functionality. If you want to use these fields, you must ensure that the required additional fields are added in ► [Data Management](#) ► [Import Data](#) ►.

Note

For all options, the same fields are proposed in both entities, *Person* and *Company*.

After you have enhanced the structures or activated pre-defined fields, you have to perform the following steps:

1. Get the metadata information of the OData service. Enter the following URL in your web browser:
`https://<server>:<port>/sap/opu/odata/sap/CUAN_BUSINESS_PARTNER_IMP_SRV/$metadata`.
Replace `<server>` and `<port>` with the corresponding data of the system where you did the enhancements.
When you process the URL you will get the metadata of the OData service including the enhancement fields.
2. Copy the metadata information in a text editor (such as Notepad) and save the data with file extension `*.EDMX`.
3. Convert the EDMX file into a XSD. For more information, see [Converting EDMX into XSD Format \[page 53\]](#).
4. Extend the mapping `COD_CUAN_Business_Partner_Replicate_Bulk` in SAP NetWeaver Process Integration (PI), or SAP HANA Cloud Integration (HCI). For more information, see:

Posting Instructions

[Extending Mapping in SAP NetWeaver Process Integration \(PI\) \[page 59\]](#), or [Extending Mapping in SAP HANA Cloud Integration \(HCI\) \[page 67\]](#).

3.1.2 Extending the Business Document OData Service in SAP Hybris Marketing

Within the OData service for importing business document information, `CUAN_BUSINESS_Document_IMP_SRV`, the entities *BusinessDocument*, *Person*, *Company*, and *Product* can be extended. It is not possible to extend the entity *ImportHeader*.

You can extend entities as follows:

- *Person/Company*:
 - Fields that should only be available in an SAP Hybris Marketing interaction contact have to be added to the structure `INCL_EEW_CUAN_CE_IC_ROOT` with `SE11`.
 - Fields which should be available in an SAP Hybris Marketing interaction contact, and interaction have to be added to the structure `INCL_EEW_CUAN_CE_IA_IC` with `SE11`.
- *BusinessDocument*: All extension fields have to be added to structure `INCL_EEW_CUAN_CE_IA_IC` with `SE11`
- Besides the extension include you can use one or multiple of the predefined fields which are provided by standard functionality. If you want to use these fields, you must ensure that the required additional fields are added in ► [Data Management](#) ► [Import Data](#) ►.

After you have enhanced the structures, or activated predefined fields, you have to proceed as follows:

1. Get the metadata information of the OData service. Enter the following URL in your Web browser:
`https://<server>:<port>/sap/opu/odata/sap/CUAN_BUSINESS_DOCUMENT_IMP_SRV/$metadata`
Replace `<server>` and `<port>` with the corresponding data of the system where you did the enhancements. When you process the URL you will get the metadata of the OData service including the enhancement fields.
2. Copy the metadata information in a text editor (e.g. Notepad), and save the data with file extension `*.EDMX`.
3. Convert the EDMX file into a XSD file. For more information, see [.Converting EDMX into XSD Format \[page 53\]](#).
4. Extend the mapping `COD_CUAN_Business_Partner_Replicate_Bulk` in SAP NetWeaver Process Integration (PI), or SAP HANA Cloud Integration (HCI). For more information, see [Extending Mapping in SAP NetWeaver Process Integration \(PI\) \[page 59\]](#), and [Extending Mapping in SAP HANA Cloud Integration \(HCI\) \[page 67\]](#).

3.1.3 Converting EDMX into XSD Format

Note

The following screenshots are based on an example extension of OData `CUAN_BUSINESS_PARTNER_IMP_SRV`, whereas all the processing steps are the same, for OData service `CUAN_BUSINESS_DOCUMENT_IMP_SRV`.

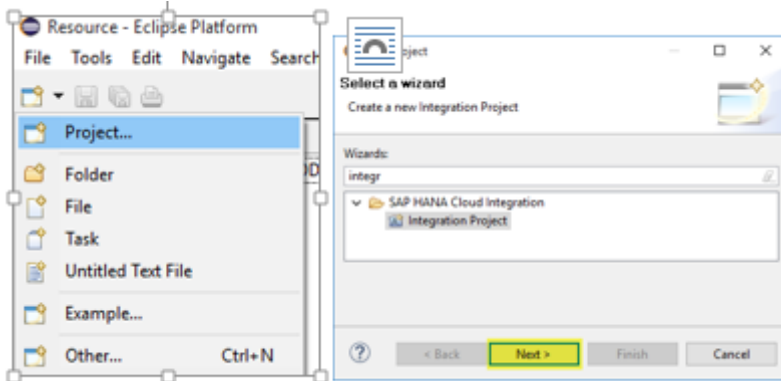
Installing Eclipse Luna

Before you can convert a EDMX into a XSD file for enhancement reasons, you have to install the Eclipse Luna tool from <https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/lunasr2> ►.

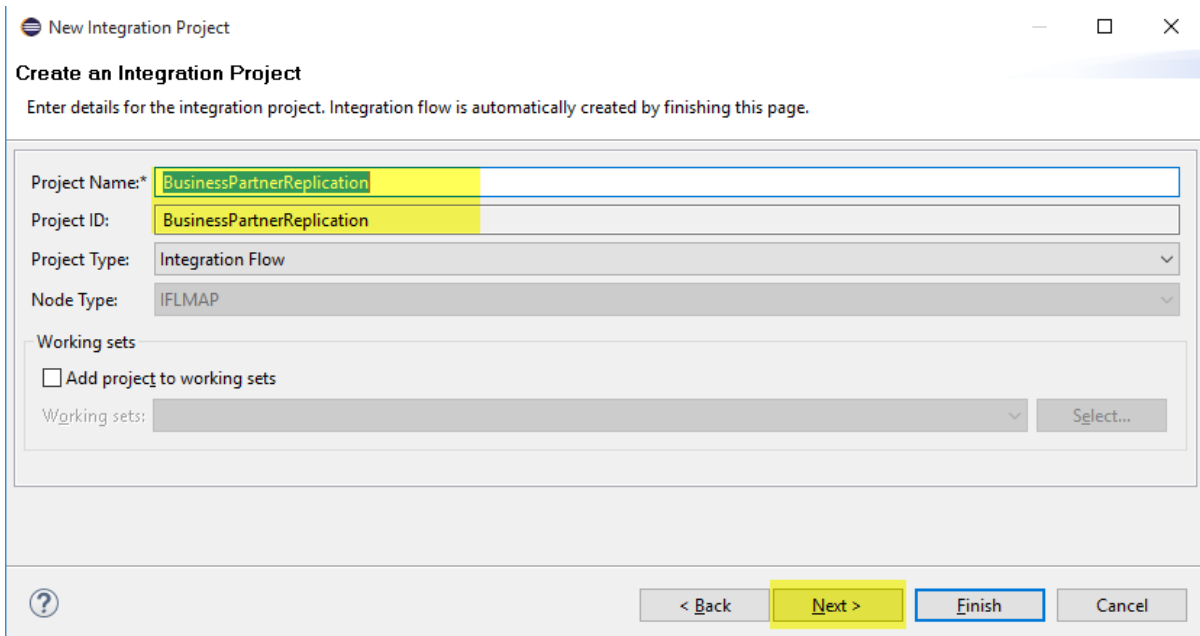
After installing the software run Eclipse Luna and go to ► [Help](#) ► [Install New Software](#) . Press the button and enter the URL <https://tools.hana.ondemand.com/luna/> in Work with. Choose [Select All](#), [Next](#) and again [Next](#). Select [I accept the terms of the license agreements](#), and choose [Finish](#).

For converting the EDMX into a XSD file, you have to do the following steps. Step 1 to step 6 must be followed only if you do not have a project in your Luna Eclipse that you could use, or if you want to create a new project. Otherwise, you can start with step 7 directly.

1. Create a new project of type Integration Project.



2. Define a project name and choose *Integration Flow* as Project Type.



3. To select a pattern for the integration flow creation, select *Point-to-Point Channel* and click *Finish*.

New Integration Project

Create an integration flow
Create an integration flow from a pattern or a template

Name:* BusinessPartnerReplication

Location:* /BusinessPartnerReplication/src/main/resources/scenarioflows/integrationflow/ Browse...

Categories

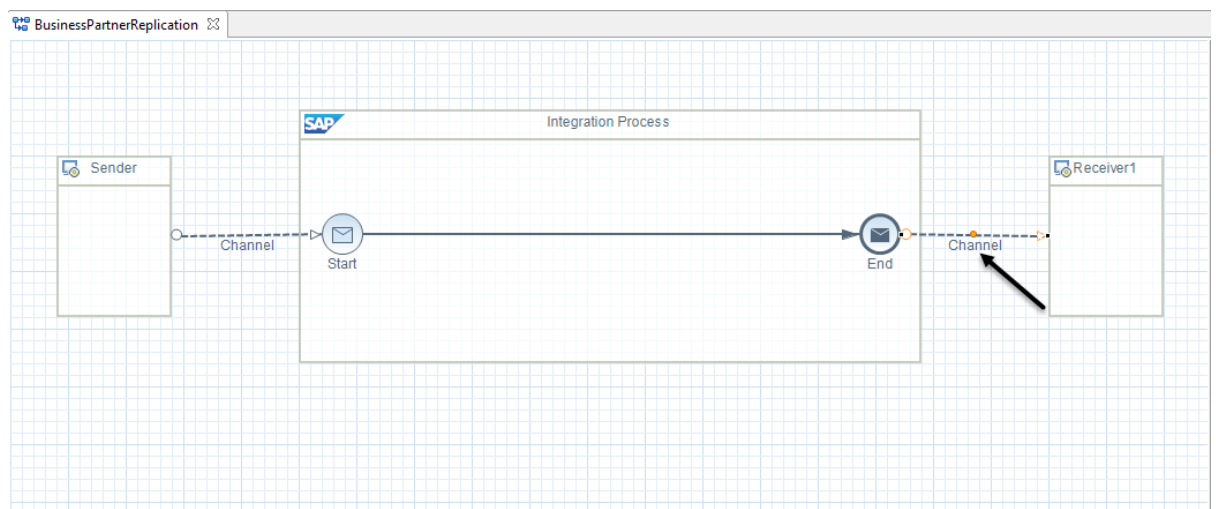
- Enterprise Integration Patterns
- SAP Defined Templates
- User Defined Templates

Select a pattern to create an integration flow

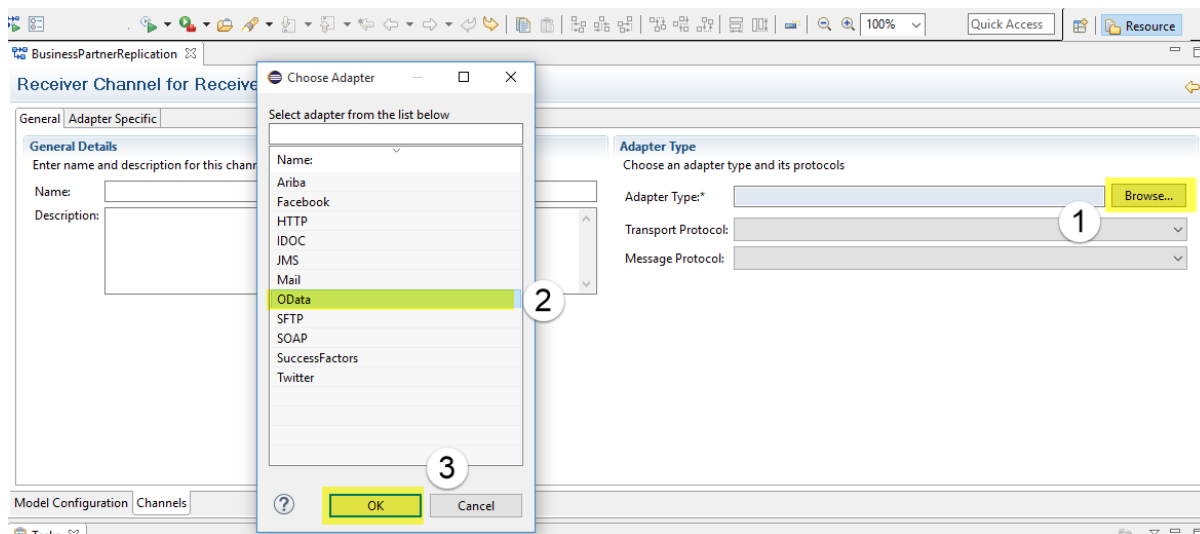
- Point-to-Point Channel**
Ensures that messages are sent only to one receiver
- Content Based Router (Multiple Receivers)
Contains one sender and multiple receivers. It examines the content of incoming message and routes it to the correct receiver.

? < Back Next > Finish Cancel

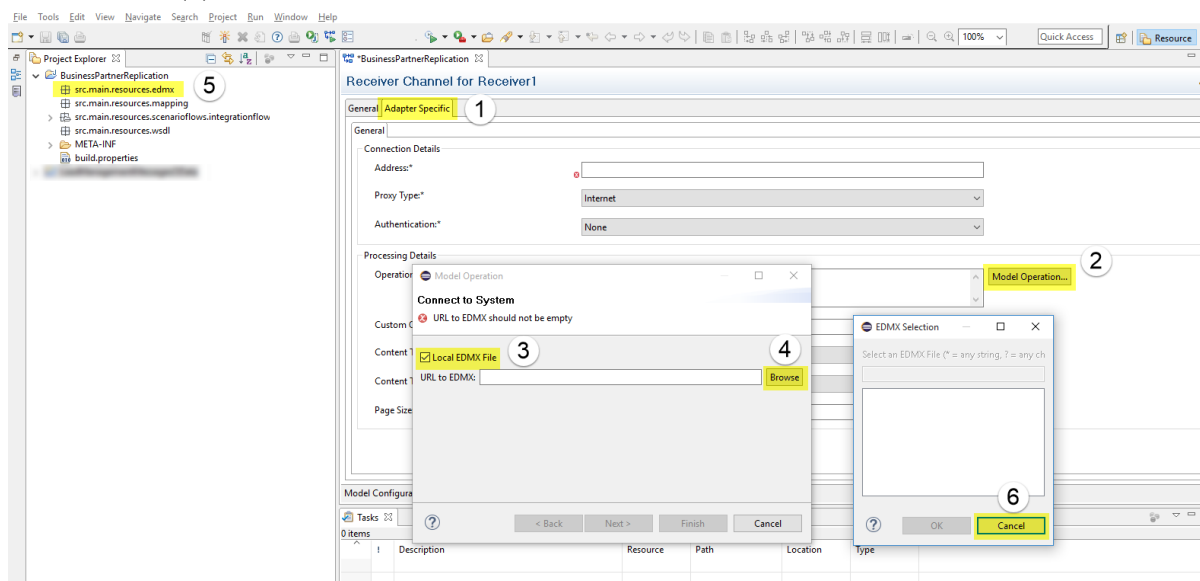
4. In the integration flow, double click *Channel*.



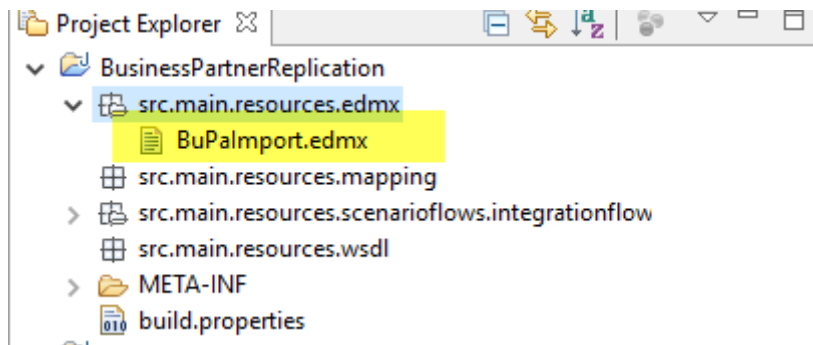
- Browse to choose *OData* as adapter, and press *OK*.



- On tabpage *Adapter Specific* (1), click *Model Operation* (2). Choose *Local EDMX File*, and *Browse*. The system displays a new folder named `src.main.resources.edmx` in your project (5), in the Project Explorer. Choose *Cancel* (6).



- Copy the OData metadata EDMX file that you have created in advance, into the new folder `src.main.resources.edmx` by drag and drop.

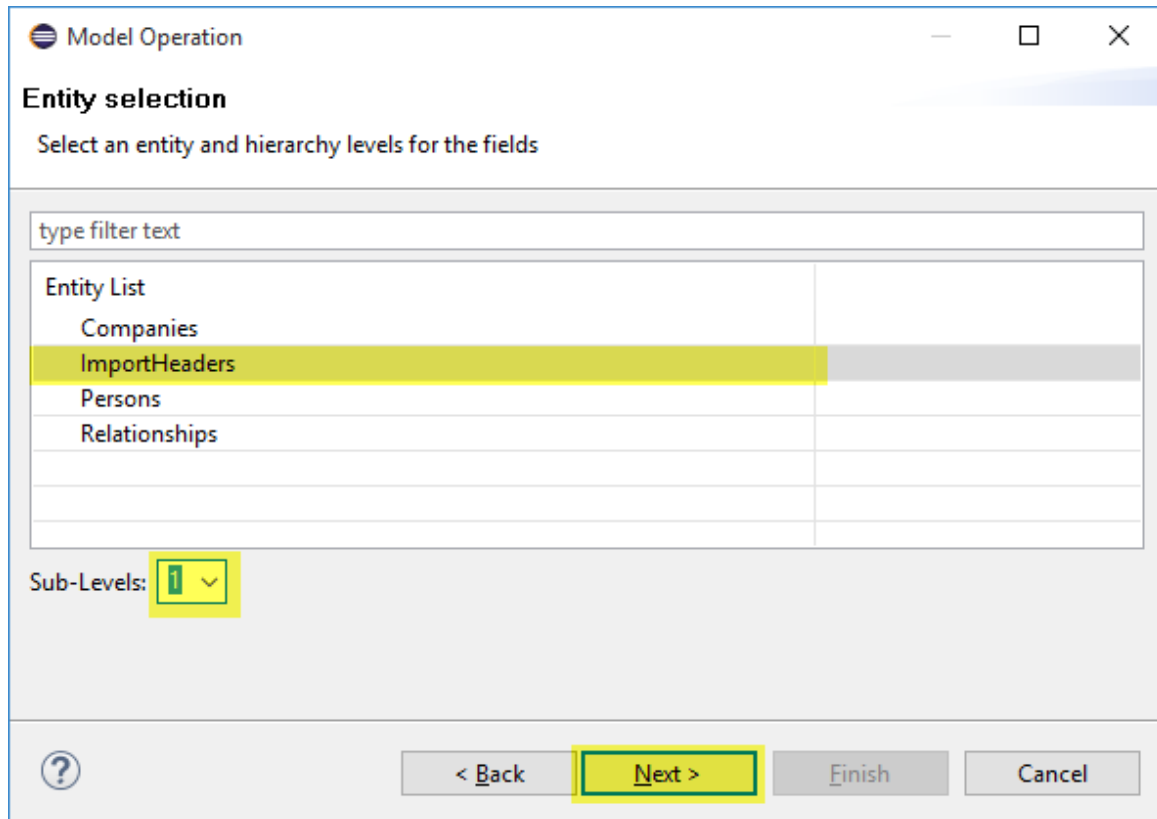


8. Repeat step 6, click *Model Operation* (2), and choose the EDMX file. The field *URL to EDMX* contains the location of the EDMX file. Choose *Next*.

The screenshot shows a dialog box titled "Model Operation" with standard window controls (minimize, maximize, close). Below the title bar, the section "Connect to System" is displayed with the instruction "Enter the details of the System to connect". A checkbox labeled "Local EDMX File" is checked. Below this, the "URL to EDMX:" field contains the path "/BusinessPartnerReplication/src/main/resources/edmx/BuPalImport.edmx", which is highlighted with a yellow background. To the right of the text field is a "Browse..." button. At the bottom of the dialog, there is a row of buttons: a help icon (?), "< Back", "Next >" (highlighted with a yellow border), "Finish", and "Cancel".

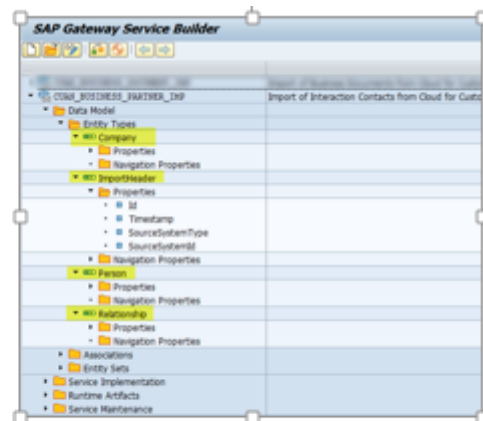
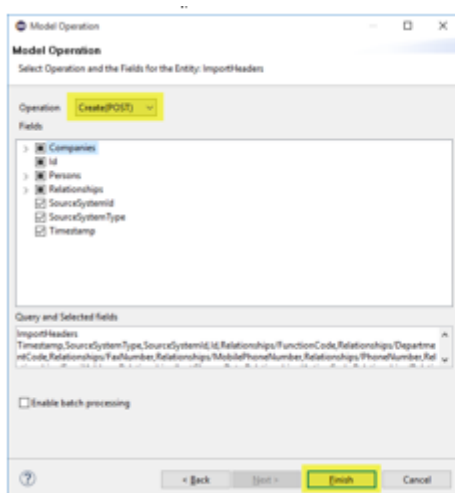
9. This step differs between the two OData services, but nevertheless you should call in parallel for both services transaction SEGW in the ABAP backend.
- CUAN_BUSINESS_PARTNER_IMP_SRV

In *Model Operation*, select *ImportHeaders*, and choose *Sub-Levels "1"*.



In *Model Operation*, select operation *Create(POST)*.

Select all properties, which belong to the different entities by standard. Therefore it is very helpful to open the OData service CUAN_BUSINESS_PARTNER_IMP_SRV in parallel in the SAP Gateway Service Builder (transaction SEGW) in the backend. As the ImportHeader is the "Root" entity, its fields are listed directly in the popup, and not below an entity named *ImportHeader*. Besides selecting the standard fields, you are now also able to select the extension fields you want, for the different entities

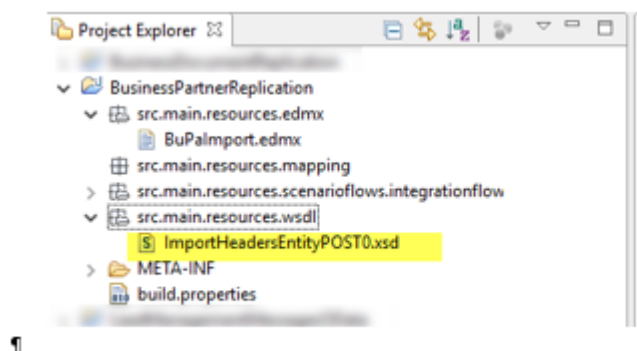


- CUAN_BUSINESS_DOCUMENT_IMP_SRV

In *Model Operation*, select *ImportHeaders* and enter *Sub-Levels "9"*.

Repeat step 2 for CUAN_BUSINESS_DOCUMENT_IMP_SRV, and call SEGW for OData service CUAN_BUSINESS_DOCUMENT_IMP_SRV

- After you have selected all properties as described in step 9 choose Finish. An XSD file is created. The file is located in folder src.main.resources.wSDL of your project.



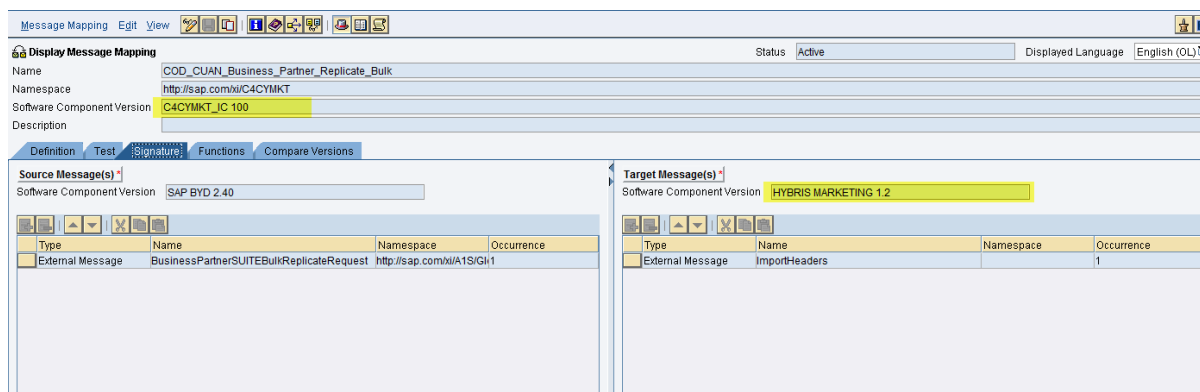
3.1.4 Extending Mapping in SAP NetWeaver Process Integration (PI)

Note

The following screenshots are based on an example extension of OData service CUAN_BUSINESS_PARTNER_IMP_SRV, whereas all the processing steps are the same, for OData service CUAN_BUSINESS_DOCUMENT_IMP_SRV.

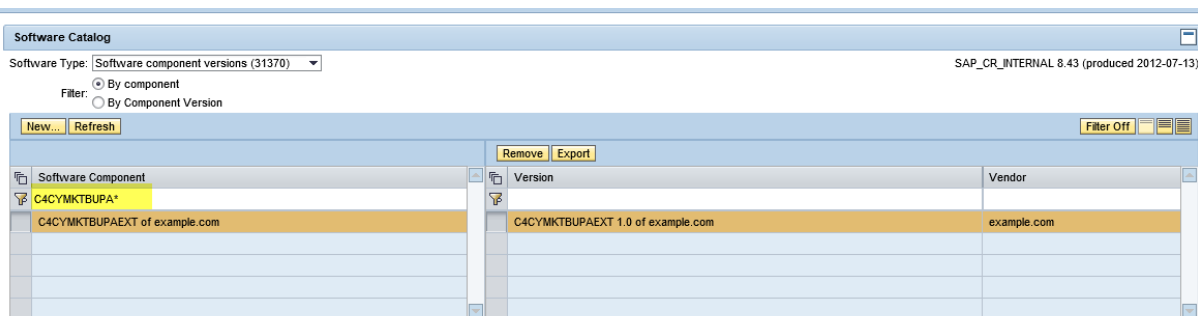
To extend the mapping in SAP NetWeaver PI proceed as follows:

- From the SAP NetWeaver PI start page open the [Enterprise Services Builder](#).
- Open the mapping that you want to extend. The following figures show the extension of the mapping COD_CUAN_Business_Partner_Replicate_Bulk (C4C Business Partner Outbound to SAP Hybris Marketing OData service CUAN_BUSINESS_PARTNER_IMP_SRV). All mappings related to CUAN_BUSINESS_PARTNER_IMP_SRV and CUAN_BUSINESS_DOCUMENT_IMP_SRV can be found in software component C4CYMKT_IC, whereas only those are relevant which starts with COD_CUAN*.
- On the [Signature](#) tabpage, note down the software component versions that requires extension. For our example these are: C4CYMKT_IC 100 and HYBRIS MARKETING 1.2.



- Open the System Landscape Directory (SLD) from the SAP NetWeaver Process Integration (PI) start page.

5. Select [Products](#) in the section [Software Catalog](#).
6. Choose [New](#) to create a new product.
7. For the action type, select [Create a new product and version](#).
8. Under [Product](#), enter the product details:
 1. Product Name: <the extension product name> (in this example C4CYMKTBUPAEXT was chosen because of extending the mapping of the Business Partner Replication)
 2. Product Vendor: <customer namespace> (such as example.com)
 3. Product Version: 1.0
9. Under [Product Instance](#), provide the Instance Name, and choose [Next](#).
10. Under [Create Software Component and Version](#), enter the software component.
 - Name: <extension software component name>
 - Version: 1.0
11. Navigate to the SLD home screen, and select [Software Components](#) from the [Software Catalogs](#) section.
12. Search for the software component that was created in the previous steps.
13. Under [Dependencies](#), select [Define Prerequisite Software Component Versions](#)



The screenshot shows the SAP Software Catalog interface. At the top, there's a header bar with 'Software Catalog' and a search icon. Below it, a dropdown menu shows 'Software Type: Software component versions (31370)'. To the right, it says 'SAP_CR_INTERNAL 8.43 (produced 2012-07-13)'. Below the dropdown, there are radio buttons for 'Filter: By component' (selected) and 'By Component Version'. There are buttons for 'New...', 'Refresh', 'Remove', 'Export', and 'Filter Off'. The main table has two columns: 'Software Component' and 'Version'. The first row shows 'C4CYMKTBUPA*' under 'Software Component' and 'C4CYMKTBUPAEXT 1.0 of example.com' under 'Version'. The second row shows 'C4CYMKTBUPAEXT of example.com' under 'Software Component' and 'example.com' under 'Version'.

Software Component	Version	Vendor
C4CYMKTBUPA*	C4CYMKTBUPAEXT 1.0 of example.com	example.com
C4CYMKTBUPAEXT of example.com		

View Products and Software Components

Software Catalog

Software Type: Software component versions (31373)

Filter: ☒ By component ☐ By Component Version

New... Refresh

Software Component	Version
C4CYMKTBUA*	
C4CYMKTBUAEXT of example.com	C4CYMKTBUAEXT 1.0 of example.com

Remove Export

C4CYMKTBUAEXT 1.0 of example.com

General Products **Dependencies** Support Packages Release Compatibility Release History

Context: InstallationTime

Define Prerequisite Software Component Versions Remove

Prerequisite Software Component Versions
--

14. Search for the first software component that you want to extend (see step 3), and click on [Define as Prerequisite Software Component](#).

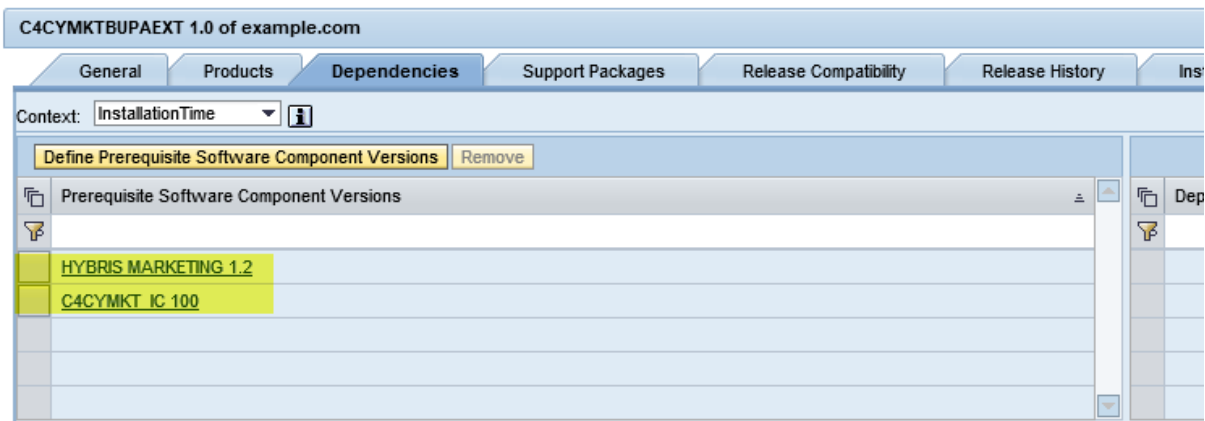
Define Prerequisite Software Components for C4CYMKTBUAEXT 1.0 of example.com

Context: InstallationTime

Software Component	Version
C4CYMKT_IC*	
C4CYMKT_IC	C4CYMKT_IC 100
XI CONTENT C4CYMKT_IC	XI CONTENT C4CYMKT_IC 100

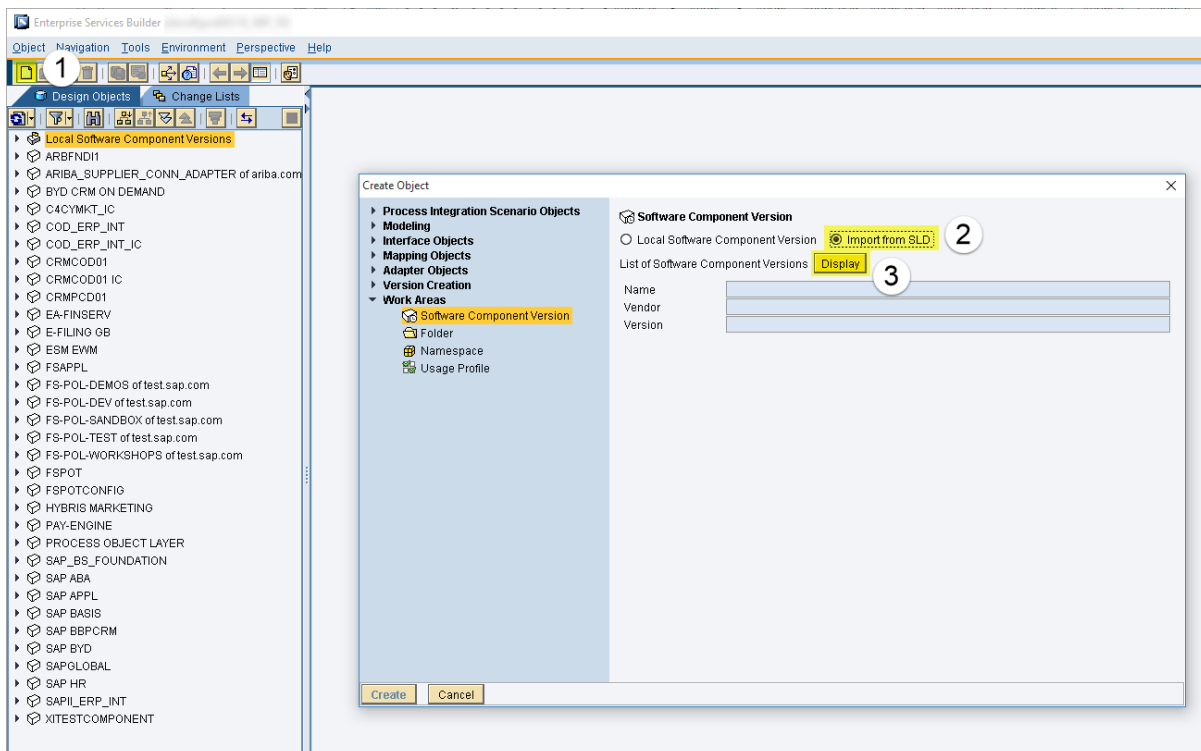
Define as Prerequisite Software Components Cancel

15. Repeat step 13 and 14 for the second software component from step 3.



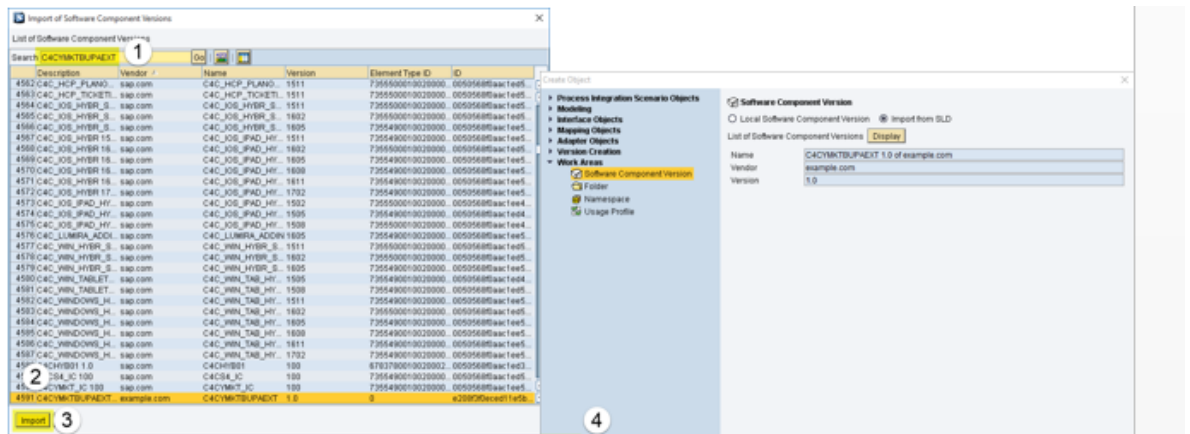
16. Navigate on the SLD home screen and open the Enterprise Service Builder.

17. Choose *Create* (1) and import (2) the software component version that was created for extension by clicking *Display*.

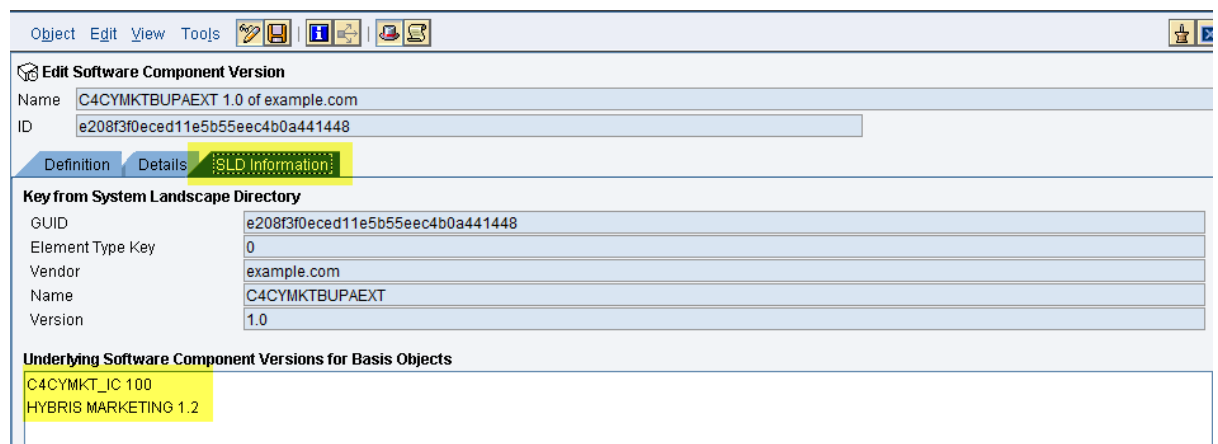


18. In the search results (1) for the software component you want to import C4CYMKTBUAEXT), select the software component (2), and choose *Import* (3).

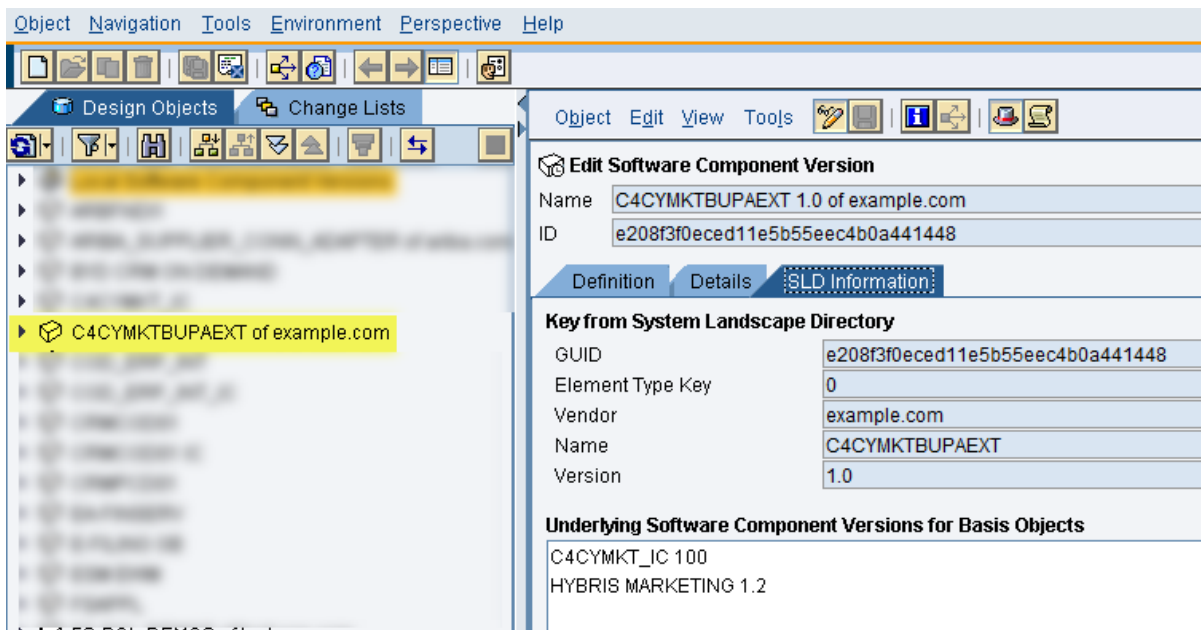
On the following popup choose *Create* (4).



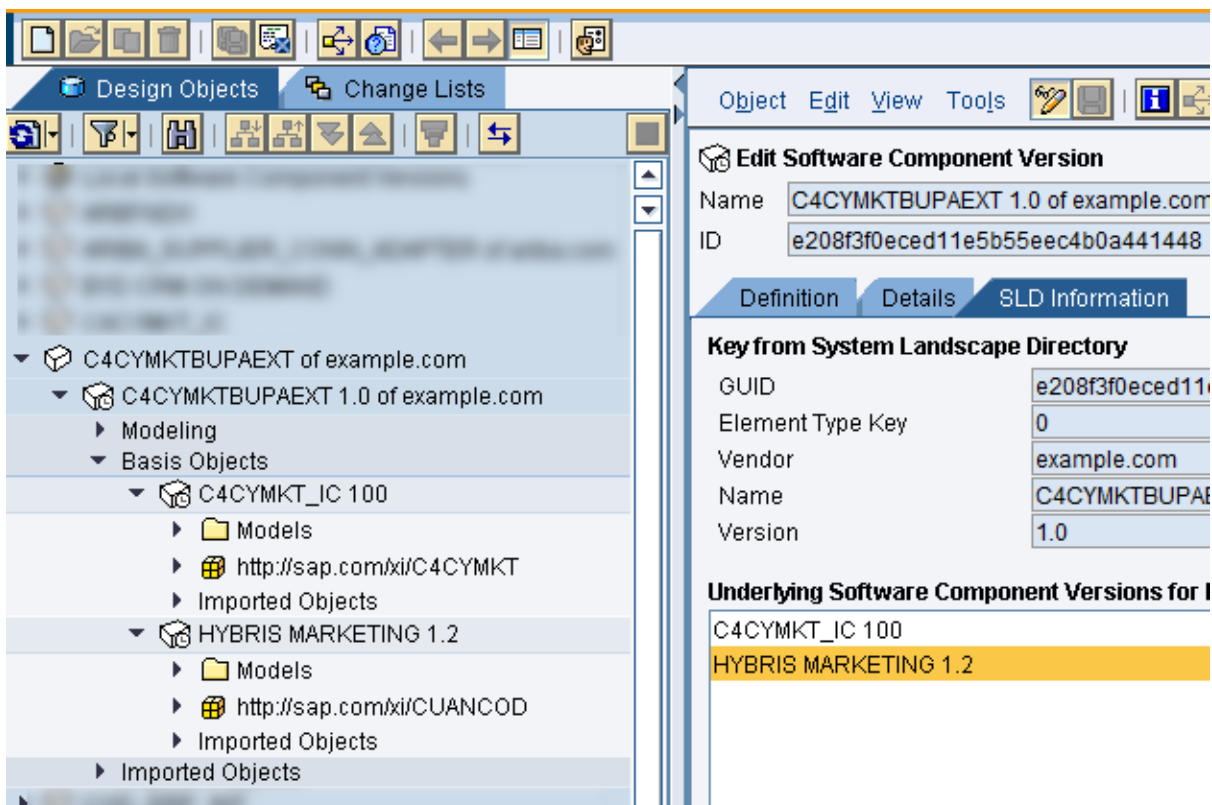
19. In *Edit Software Component Version*, on the *Definition* tabpage, select Original language *English* for the new software component version.
20. Under *SLD Information*, verify if the underlying software components contain the components noted in step 3, C4CYMKT_IC 100 and HYBRIS MARKETING 1.2



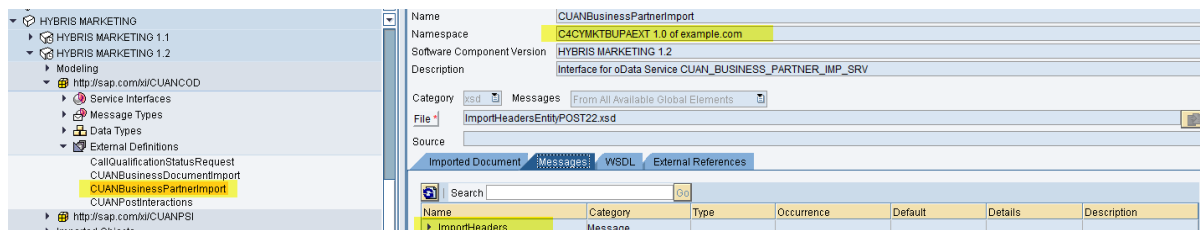
21. After saving, the software component version appears on the left tree view.



22. Expand the software component to display the basic objects containing the prerequisites software component versions.

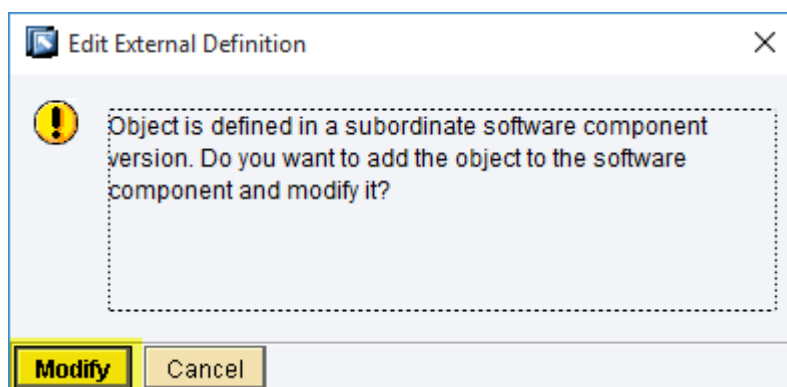


23. Open the external definition that contains the service you want to extend, `CUANBusinessPartnerImport`

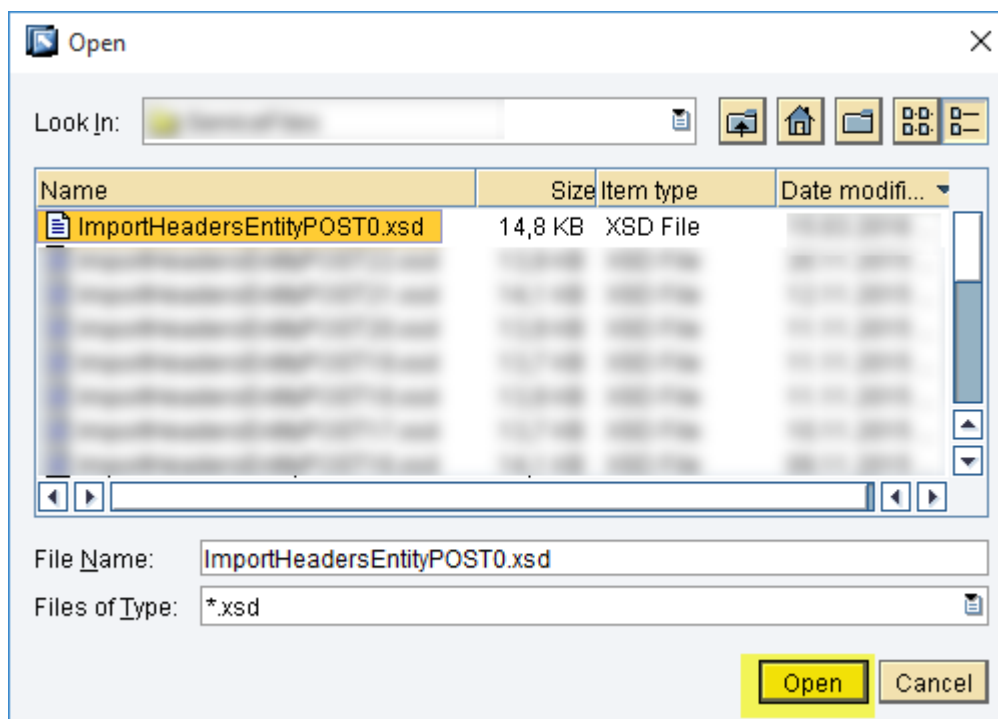


If you want to extend the `CUAN_BUSINESS_DOCUMENT_IMPORT_SRV` OData service, you have to choose `CUAN_BusinessDocumentImport`

24. Switch to the Edit mode and import the *.XSD file, which you have created from the OData service metadata call and that contains the extension fields.



1. Choose *Modify*.
2. Change the Category to *XSD*, and choose *Import*.



3. Save the external definition.

External Definition Edit View Tools

Status: In Process Displayed Language: English (OL)

Name: CUANBusinessPartnerImport

Namespace: http://sap.com/xi/CUANCOD

Software Component Version: HYBRIS MARKETING 1.2

Description:

Category: xsd Messages: From All Available Global Elements

File: ImportHeadersEntityPOST0.xsd

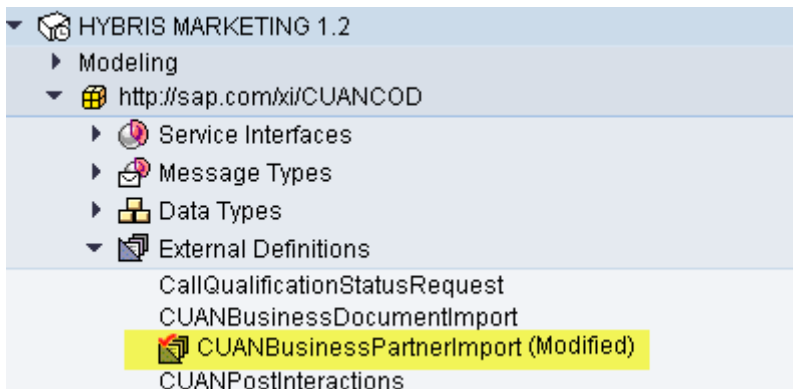
Source:

Imported Document Messages WSDL External References

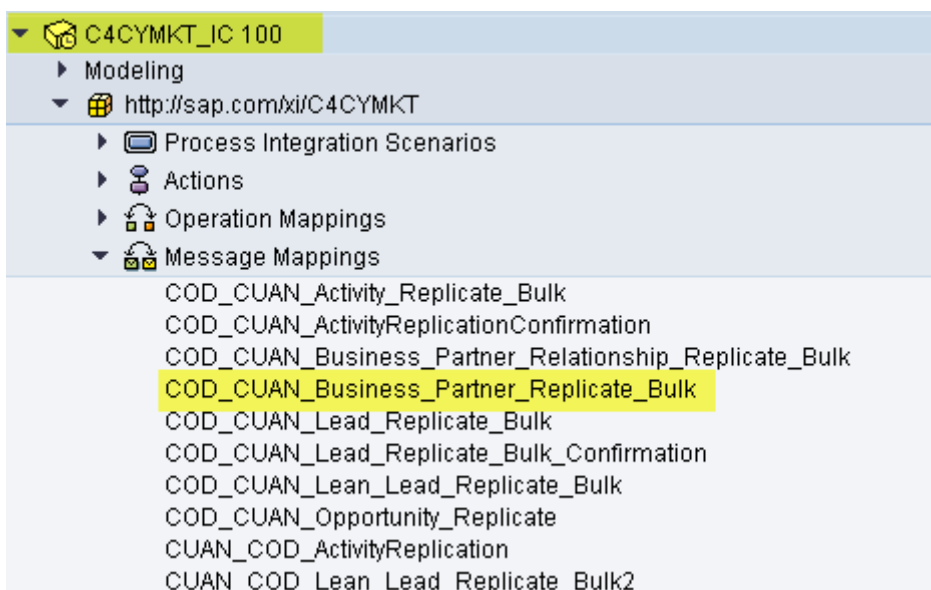
Find: Execute

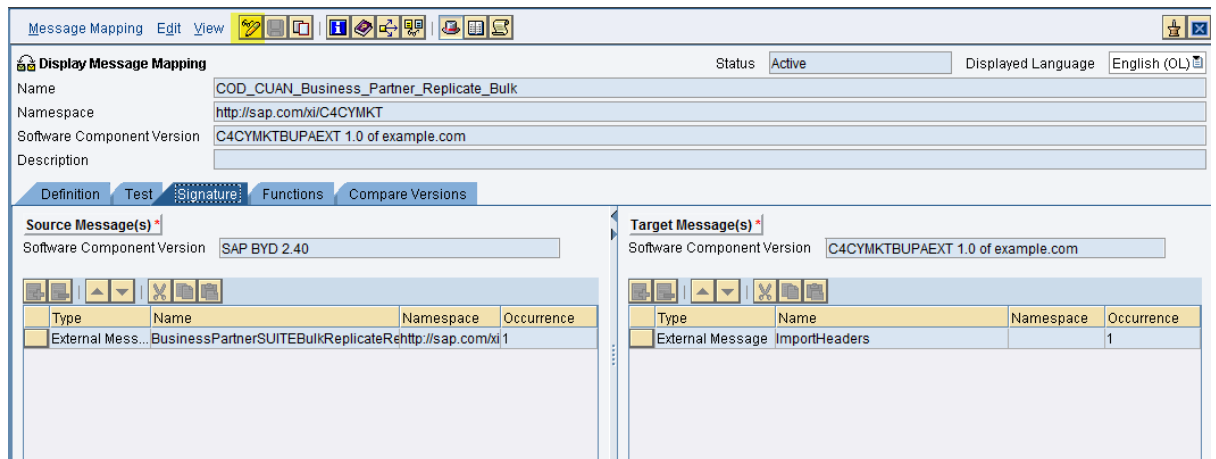
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" attributeFormDefault="unqualified" elementFormDefault="qualified">

25. In the tree view, the object is marked as *Modified*



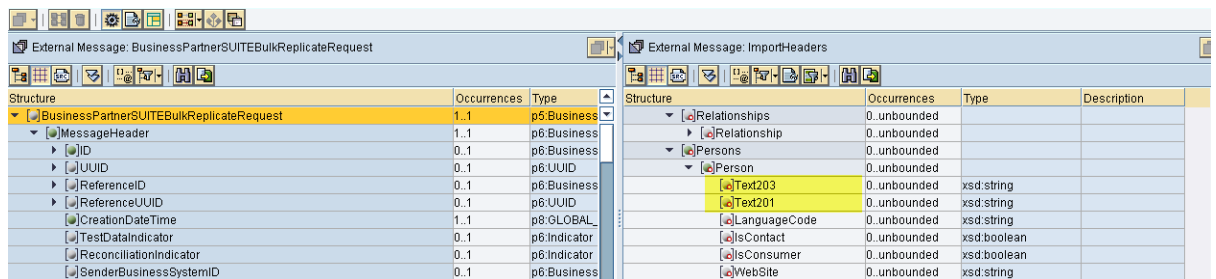
26. Open the message mapping COD_CUAN_Business_Partner_Replicate_Bulk in the software component C4CYMKT_IC 100 in edit mode, and refer to the external definition.
All mappings relying to SAP Hybris Cloud for Customer / SAP Hybris Marketing integration, which are related to the two OData services CUAN_BUSINESS_PARTNER_IMP_SRV, and CUAN_BUSINESS_DOCUMENT_IMP_SRV start with COD_CUAN.





Save the mapping.

27. Switch to the **Definition** tabpage. The mapping contains the new field which can be mapped.



28. Save all your changes, and activate the objects.

29. Make sure to change the Interface Determination in **Integration Builder** to use the interface mapping from the new software component, that is, C4CCRMEXT. In case of Java only SAP PI (AEX) installations change the Integrated Configuration to use the mapping from the new software component.

3.1.5 Extending Mapping in SAP HANA Cloud Integration (HCI)

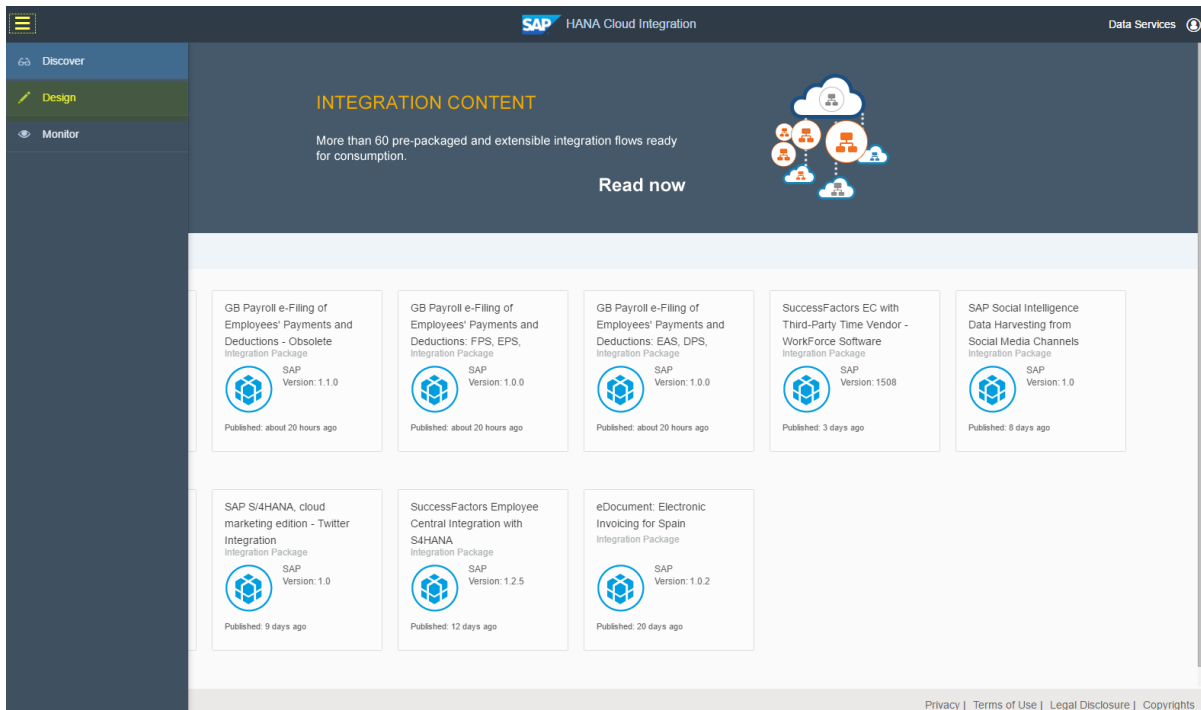
Note

The following screenshots are based on an example extension of OData service

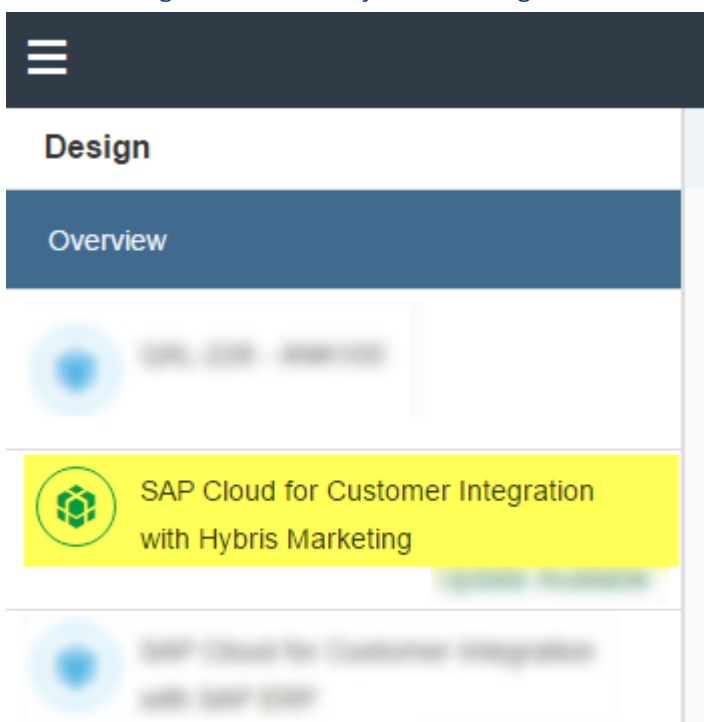
CUAN_BUSINESS_DOCUMENT_IMP_SRV, whereas all the processing steps are the same, for OData service CUAN_BUSINESS_PARTNER_IMP_SRV

For extending the mapping in SAP HANA Cloud Integration (HCI), you proceed as follows:

1. Launch the SAP HANA Cloud Integration Web UI, and choose **Design**.




- Choose the integration package that contains the iFlow which should be extended. For SAP Hybris Marketing / SAP Hybris Cloud for Customer integration, choose the integration package *SAP Cloud for Customer integration with SAP Hybris Marketing*.






- Open the iFlow project that should be extended. To extend the lead replication from SAP Hybris Cloud for Customer to SAP Hybris Marketing, the inbound processing in SAP Hybris Marketing is done by the OData service CUAN_BUSINESS_DOCUMENT_IMP_SRV.

SAP Cloud for Customer Integration with Hybris Marketing







SAP Cloud for Customer Integration with Hybris Marketing

Version: [1.0.0](#) | Owned By: [SAP](#) | Last Modified By: [SAP](#) | Mode: [Editable](#)
 Created By: [SAP](#) | Creation Date: [2020-11-18 10:00:00](#) | Last Modified Date: [2020-11-18 10:00:00](#)
 Description:
 This integration package enables you to integrate business processes between SAP hybris Marketing and SAP Cloud for Customer. The integration scope includes replication of call qualifications to SAP Cloud for Customer as Leads and replication of Lead status back to SAP hybris Marketing.


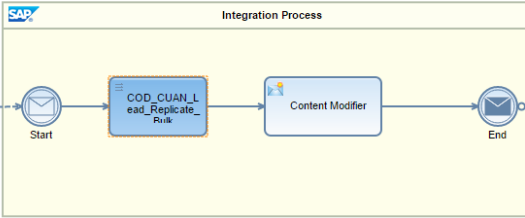





Artifacts (4)

<input type="checkbox"/>	Name	Version	Type	Actions
<input type="checkbox"/>	Replicate Activity to Hybris Marketing Created	1.0.0	Process Integration	
<input type="checkbox"/>	Replicate Lead Confirmation to Hybris Marketing Created	1.0.0	Process Integration	
<input type="checkbox"/>	Replicate Lead to Hybris Marketing Lead Replication to SAP Business Suite Created	1.0.0	Process Integration	
<input type="checkbox"/>	Value Mapping Value Mapping for Cloud for Customer and Hybris Marketing Integration Modified Update Available	Draft	Value Mapping	

4. Choose *Edit*.

← Replicate Lead to Hybris Marketing

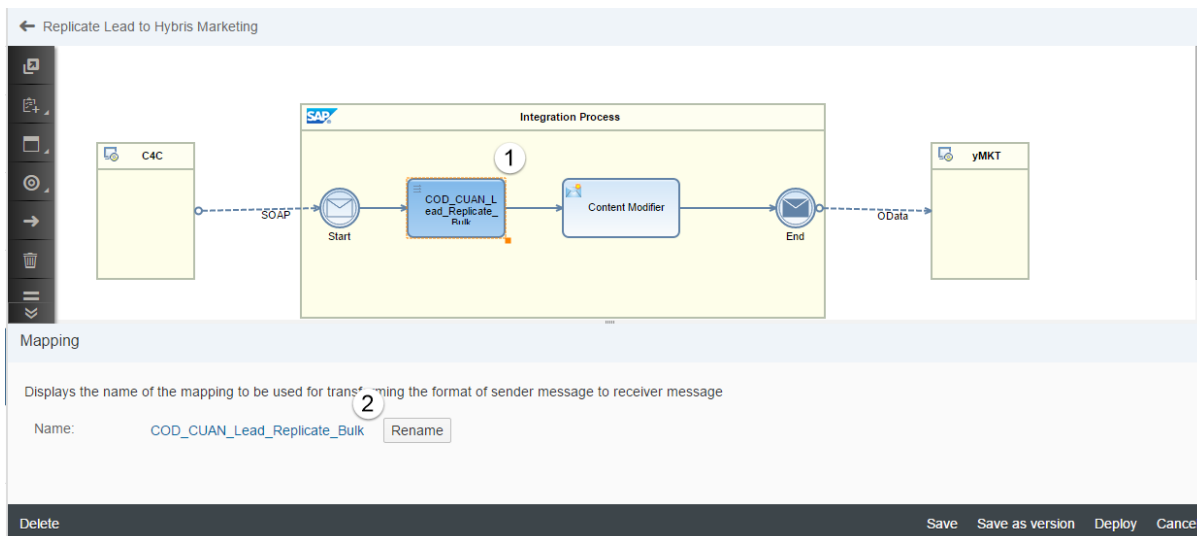
Mapping

Displays the name of the mapping to be used for transforming the format of sender message to receiver message

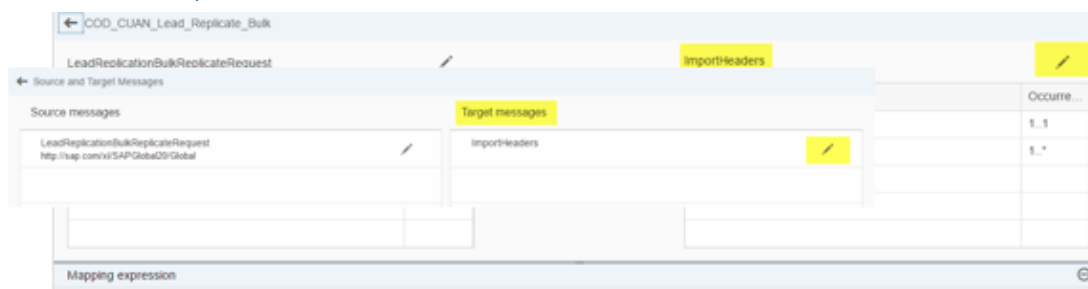
Name: [COD_CUAN_Lead_Replicate_Bulk](#)

[Delete](#)
[Edit](#)
[Deploy](#)

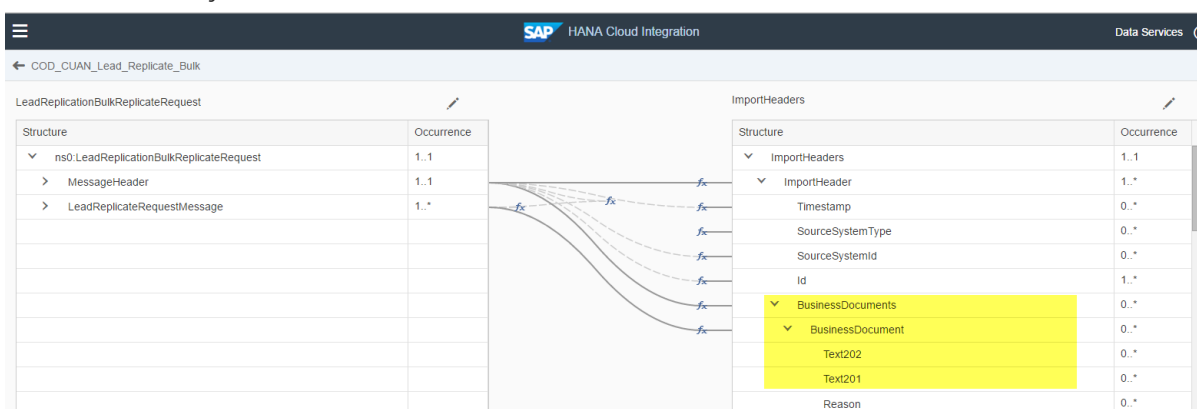
- In the iFlow, select the mapping entity (1) and click on the link (2) to open the mapping.



- In mapping editor, click the edit icon to import the file. To extend the OData service inbound message, you click the edit icon of *ImportHeaders*.



- Click the Edit icon of the *Target message* for importing the *.XSD file you have created out of the OData service metadata EDMX file.
- Choose Choose from File System and upload the *.XSD file.
- After importing the file, select the message type and click OK to accept the change in the structure.
- Expand the nodes in the mapping editor to the extended field(s), and do the mapping with the corresponding fields of the SAP Hybris Cloud for Customer outbound service.



- Save and deploy the iFlow.

3.1.6 Lead Management

In SAP Hybris Marketing, you can exchange business partner and business document data, such as leads, or opportunities with SAP Hybris Cloud for Customer. The following chapter provides you with information on how to extend the relevant services on SAP Hybris Marketing side to import additional data from SAP Hybris Cloud for Customer.

3.2 Relationship Analysis

Including Customer Specific Fields in Relationship Analysis

You can include customer specific fields in the data source used by the *Relationship Analysis* sub workset, which is a part of the *Insight* workset. Typically, you require additional fields when you have extended your ERP or CRM database tables with customer specific fields, and want to include these fields in *Relationship Analysis*. The fields you define as filter attributes for *Relationship Analysis* can also be used in other worksets, for example, in *Segmentation*.

You perform the following main tasks to include additional fields:

- Provide SAP HANA information models that contain the additional fields. You copy standard models delivered with *SAP hybris Marketing* and add the fields to the copied models.
- Check whether the SAP HANA information models you provide for the additional fields are correctly processed in the analytical data flow.
- Adapt the Easy Queries that are based on the SAP HANA information models you provide for the additional fields.
- Define the additional fields as filter attributes for *Relationship Analysis*.
- Clean up the gateway cache.

Enabling the Saving of a Personalized View for a Custom Relationship Analysis Workset

To enable the *Save View* option for **custom** *Relationship Analysis* worksets you create a page for the custom workset performing the according page administration operations. Consider the following:

- For general information about how to display and use the page administration user interface to create a page, see the Help Portal at <http://help.sap.com/netweaver> ► *SAP NetWeaver Platform* ► *SAP NetWeaver 7.5* ► *Application Help* ► *UI Technologies in SAP NetWeaver with SAP_UI 750* ► *SAP NetWeaver User Interface Services* ► *Configuration and Operation* ► *Content Administration* ► *Suite Page Builder* ► *Page Administration* ►.
- For a step-by-step description and the required page details for a custom *Relationship Analysis* workset, see section *Enabling the Saving of a Personalized View* in this guide.

3.2.1 Providing SAP HANA Information Models

Use

Use SAP HANA Studio to create custom copies of the relevant standard SAP HANA information models.

For more information about the SAP HANA information models that are consumed by *Relationship Analysis*, see *Business Content* on the help portal at <http://help.sap.com/mkt>.

Note

For the copy procedure, make sure you have the following privileges:

- REPO.READ (source package)
- REPO.MAINTAIN_NATIVE_PACKAGES (root package)
- REPO.EDIT_NATIVE_OBJECTS (root package)

Note

The performance of the application depends on the performance of the SAP HANA information models and SQLscript procedures you provide. Familiarize yourself with the modeling guidelines to ensure that your information models and procedures are as performant as possible.

For more information about the modeling, see the *SAP HANA Modeling Guide*, and the *SAP HANA Developer Guide* at http://help.sap.com/hana_platform. In addition, consider the *SAP HANA Performance Analysis Guide* for information about how to identify and resolve performance issues of your SAP HANA database.

Note

Make sure you use the correct format when you indicate the path and the name of SAP HANA information models, or SQLscript procedures:

- Lower case for the path, for example, `sap.hana-app.cuan.cpred.datafoundation`
- Upper case for the name of the model, or procedure, for example, `CA_CPRED_CRM_DS_PROD_AFFIN`
- Separate the path from the name by a slash `</>`.

Proceed as follows to copy the relevant models:

1. Locate the standard SAP HANA information models delivered for *Relationship Analysis*. For example, you can find the calculation view *Relationship Analysis with Target Group Integration* (CA_AI_RELATIONSHIP_MONITOR_TG) under **sap > hana-app > cuan > ai > Calculation Views**.
2. Check the interdependencies between the models that are relevant for the customer specific fields you want to add. We recommend to copy all dependent models to avoid any cross reference from your custom specific models to the standard models. For example, the *Relationship Analysis with Target Group Integration* calculation view processes data that is provided by the calculation view CA_AI_CUST_REL_MONITOR, and by the attribute view AT_CUSTOMER_ATTRIBUTE.
For each model, you can use the *Auto Documentation* function in SAP HANA Studio to check the interdependencies with other models.
You can copy each relevant model individually, or you can first select the models and then copy them all at once.

3. Create a package (folder) that can be used as the target root package when you copy the standard SAP HANA information models. We recommend to restrict the naming of the package to 3 characters, such as **ext**.

i Note

If you need to include additional custom fields later on, use the same custom root package, for example, **ext**. You cannot use different custom root packages.

4. In SAP HANA studio ► **Quick Launch** ►, choose **Mass Copy**. For the package mapping, add the root package for the source, which is **sap** for the standard models. In addition, select your target root package. Using the **Mass Copy** function ensures that the data flow from your custom models to **Relationship Analysis** and other worksets is initially enabled.
5. Select all models that are relevant for the customer specific fields you want to include. Consider all interdependencies. Choose **Next** to start the copy process.
6. Check the package structure that is created as a copy of the standard package. Check whether all relevant models are available in the copied package.

Once you have copied all relevant SAP HANA information models to your custom package, proceed as follows to add your customer specific fields:

1. Typically, you first add the customer specific fields in the attribute views referencing the database tables that include your customer specific fields. For example, the attribute view **AT_CUSTOMER_ATTRIBUTE** references the database table **KNA1** of your ERP system. In SAP HANA Studio, open the attribute view. Add the field **ZZTERRITORY** to the output. Save and validate the changes. Activate the changed attribute view.
2. Subsequently, add the customer specific fields in all relevant models. You can start from the top calculation view, for example, from **Relationship Analysis with Target Group Integration** (**CA_AI_RELATIONSHIP_MONITOR_TG**), using the graphical representation of the models consumed by the top calculation view.
3. Add the customer specific fields also in the top calculation view. Map each source field to the target. Finally, include the new fields in the output of the model. Save, validate, and activate the model.








3.2.2 Validating Customer Specific Fields in the TransientProvider

Proceed as follows to check whether the customer specific fields you have added to your custom SAP HANA information models are correctly processed in the corresponding TransientProvider:

1. Start transaction **Display TransientProvider Preview for Operational Data Provider** (**RSRTS_ODP_DIS**).
2. Use the input help for **ODP Content** to select **SAP HANA Analytic Views**.
3. Use the input help for **ODP Name** to select the custom package of SAP HANA information models you have created. Check the **Software Component** list to find your custom package.
4. Choose **Execute** to display the list of fields that are contained in the TransientProvider for your custom package.
5. In the list view, check whether the customer specific fields are included.
6. In addition, choose **Standard Query** to check whether the new fields are included in the query.

Note

If you cannot find your customer specific fields, use the option *Read Without Buffer*. With this option, the transaction displays the TransientProvider content as it currently reads in the SAP HANA information models. If the current content as read from the models does not show your customer specific fields, check the status of the models you have adapted for the additional fields.

For more information about the objects involved in the analytical data flow, see the [Dataflow Overview](#) on the help portal at <http://help.sap.com>  [SAP Business Suite](#)  [SAP HANA Innovations for SAP Business Suite](#)  [Applications powered by SAP HANA](#)  [SAP hybris Marketing 1.0](#)  [Application Help](#)  [Business Content for SAP hybris Marketing](#) .

3.2.3 Adapting the Relevant Easy Queries

Use the *BEx Query Designer* as follows to adapt all relevant Easy Queries to include the customer specific fields:





1. For the *Search in InfoAreas*, use the input help to select *Unassigned Nodes*.
2. In the list of unassigned nodes, select the custom SAP HANA information model providing customer specific fields (amongst the standard fields), for example, the calculation view *Relationship Analysis with Target Group Integration*. Choose *Open*.
3. The list shows all queries processing business data that is provided by the SAP HANA information model you have selected. Check which queries should include the customer specific fields.
4. In each of the relevant queries, add the customer specific fields to the rows of the query.

Note

If you cannot find your customer specific fields in the *BEx Query Designer*, use transaction *Shared Memory* (SHMM) to clear the shared memory.

3.2.4 Defining Customer Specific Fields as Filter Attributes

Define the customer specific field attributes to make them available as attributes you can use for filtering in the *Relationship Analysis* sub workset.

You define the filter attributes in Customizing for *SAP hybris Marketing* under  [Customer Value Intelligence](#)  [Settings for Relationship Analysis and Stratification](#)  [Define Available KPIs for Relationship Analysis and Stratification](#) .

Proceed as follows:

1. In the Customizing activity *Define Available KPIs for Relationship Analysis and Stratification*, under *Data Source Alias*, select *RELANAEASYQUERY*. Choose *Define Filter*.
2. Create a new entry for each Easy Query you have adapted to include the customer specific fields. Under *OData Property Name*, enter the technical name of the query. Use the prefix A in front of the technical name. As a result, the entry should read: A<technical name of the query>.

For more information about additional customizing options, see the help topic assigned to the customizing activity.

3.2.5 Cleaning up the Cache

Finally, you clean up the gateway cache to ensure the customer specific fields are available on the UI. Use the following transactions:

- Clean up of Model Cache (/IWBEF/CACHE_CLEANUP)
- Clean up of Model Cache (/IWFND/CACHE_CLEANUP)

i Note

Use /n to start the transactions.

For both transactions, use the option *Cleanup Cache for all Models*.

3.2.6 Enabling the Saving of a Personalized View

To enable the saving of a personalized view for a custom *Relationship Analysis* workset use the page administration UI to create a page. Perform the following steps:

1. In the backend system, create a specific transport request (before you create a page).
For the details about how to create the specific transport request, and how to set up your user profile, see the Help Portal at <http://help.sap.com/netweaver> ► *SAP NetWeaver Platform* ► *SAP NetWeaver 7.5* ► *Application Help* ► *UI Technologies in SAP NetWeaver with SAP_UI 750* ► *SAP NetWeaver User Interface Services* ► *Configuration and Operations* ► *Content Administration* ► *Transporting Pages* .
The page you create (in the following step) is automatically assigned to the transport request.
2. In the page administration, choose *Create* (on the upper right hand side of the page administration UI). Enter the following values for the parameters of the page setup:
 - For the *Page ID*, enter the application alias of the custom workset (as defined in the according PFCG role) adding the following prefix and suffix: **HPA_CUAN_<application alias>_P**.
For more information about how to add a custom workset to SAP Hybris Marketing, see chapter 2 *Additional Applications or Facets* in this guide.
 - *Title*: Enter a title for the page.
 - *Catalog*: **/UI2/CATALOG_ALL**
3. Choose *OK* to create the page. The newly created page enables the saving of a personalized view for the custom *Relationship Analysis* workset.

3.3 Displaying a Key Figure or Characteristic in a Chart

To display a new key figure or characteristic of your source system in a *SAP hybris Marketing* chart (such as the charts used in *Relationship Analysis*), you need to copy the corresponding SAP HANA information model, enhance

this copy with the required key figure/characteristic, adapt the corresponding easy query, and check whether these changes have been prompted to the [SAP hybris Marketing](#) front end.

This documentation describes the detailed steps necessary to display a new key figure/characteristic on the user interface.

Copy and Enhance SAP Information Model

First you copy the required information model in the SAP HANA modeler, a SAP HANA studio perspective that allows technical users to create and adapt SAP HANA information models. Once you have copied the required information model, you enhance the model with the required key figures or characteristics.

Note

The performance of the application depends on the performance of the SAP HANA information models and SQLscript procedures you provide. Familiarize yourself with the modeling guidelines to ensure that your information models and procedures are as performant as possible.

For more information about the modeling, see the [SAP HANA Modeling Guide](#), and the [SAP HANA Developer Guide](#) at http://help.sap.com/hana_platform. In addition, consider the [SAP HANA Performance Analysis Guide](#) for information about how to identify and resolve performance issues of your SAP HANA database.

Note

Make sure you use the correct format when you indicate the path and the name of SAP HANA information models, or SQLscript procedures:

- Lower case for the path, for example, `sap.hana-app.cuan.cpred.datafoundation`
- Upper case for the name of the model, or procedure, for example, `CA_CPRED_CRM_DS_PROD_AFFIN`
- Separate the path from the name by a slash `</>`.

Enhance Corresponding Query

To make sure that the required key figure/characteristic in the enhanced SAP HANA information model is displayed in the required [SAP hybris Marketing](#) chart, you need to enhance the corresponding query in the BEx Query Designer.








1. To do so, open the BEx Query Designer and connect it to your AS ABAP system.
2. Choose the [Open](#) menu item in the [Query](#) menu.
3. In the following dialog box, choose [Find](#) in the [Search in](#) section, and search for the query that corresponds to the SAP HANA information model that you have enhanced in the SAP HANA modeler.
4. You find the newly added key figure/characteristic in the [Info Provider](#) column.
5. Choose the [Rows/Columns](#) tab in the center of the BEx Query Designer window.
6. If you want to add a key figure, drag and drop it from the [Info Provider](#) column to the [Columns](#) area.
7. If you want to add a characteristic, drag and drop it from the [Info Provider](#) column to the [Rows](#) area.
8. Save the query.

For more information about the BEx Query Designer, see the SAP Help Portal at <http://help.sap.com/nw75>.

► [Application Help](#) ► [Function-Oriented View](#) ► [SAP Business Warehouse](#) ► [SAP Business Explorer](#) ► [BEx Query Designer](#) ►








Note

With regard to potential SAP updates, it is not critical to enhance the BEx query as you modify the active version of the query but not the delivery version.

For more information about the versioning concept of BW objects in general, see the SAP Help Portal at <http://help.sap.com/nw75>  [Application Help](#)  [Function-Oriented View](#)  [SAP Business Warehouse](#)  [Generic Tools and Services](#)  [Transporting BW Objects, and Creating, Delivering and Copying BI Content](#)  [BW Versioning Concept](#) .

Determine the Type of Data Supply for Relationship Analysis

You can determine the type of data supply for the *Relationship Analysis*. The *Relationship Analysis* is a subworkset of the *Insight* workset. By default, the data is supplied by an OData services that is created by the *SAP NetWeaver Gateway Service Builder* (transaction SEGW). As an alternative, the data can be supplied by an OData service that is based on an Easy Query.

Use the *SAP Customizing Implementation Guide* to determine the data source type used for *Relationship Analysis*: Choose  [Customizing - Edit Project \(transaction SPRO\)](#)  [SAP Reference IMG](#)  [SAP hybris Marketing](#)  [Insight](#)  [Settings for Relationship Analysis and Stratification](#)  [Define Available Attributes for Relationship Analysis and Stratification](#) .

For more information about the data flow in *SAP hybris Marketing*, see the application help at <http://help.sap.com>  [SAP Business Suite](#)  [SAP HANA Innovations for SAP Business Suite](#)  [Applications powered by SAP HANA](#)  [SAP hybris Marketing, Powered by SAP HANA](#)  [Business Content for SAP hybris Marketing](#)  [Dataflow Overview](#) .

Check Query Enhancement in Back End

The query you enhanced in the previous step is realized as an easy query. An easy query is a query that can be created quickly and with very little need for parameterization. The easy query ensures that it is available as a service which means that it can be displayed in an HTML5 user interface.

The easy query is regenerated automatically after the enhancement in the previous step. You can check the regeneration procedure in the easy query manager of your back-end system (transaction eqmanager).

To do so, search for your easy query in the easy query manager and check the *Change Date for Query* field. If your easy query has not been regenerated successfully, you need to regenerate it manually by selecting it and choosing the *Regenerate Easy Query* pushbutton.

After successful regeneration of the easy query, your key figure/characteristic is available in the required chart on the user interface.

3.4 Integrating SAP Lumira

See the following sections for the details of how to set up SAP Hybris Marketing to enable the call of *SAP Lumira* reports from SAP Hybris Marketing.









Note

To configure the call of an SAP Lumira report in SAP Hybris Marketing, you require the OpenDocument URL of the report. To find the OpenDocument URL, see the *SAP Lumira User Guide* at http://help.sap.com/businessobject/product_guides/lumC1/en/lumC_user_en.pdf  *Working with files in SAP Lumira*  *Getting the embeddable URL for a private story* .

Make sure the OpenDocument URL correctly includes the string `open?id=`.










Configuring the Server Connection

To enable the call of a report from the SAP Hybris Marketing, create an RFC connection in the AS ABAP system as follows:

1. Logon to your AS ABAP system, and start the *Configuration of RFC Connections* (transaction SM59).
2. Select *HTTP Connections to ABAP System*, and choose *Create*.
3. For the *RFC Destination*, enter **LUMIRA_CLOUD**.
4. On tab  *Technical Settings*  *Target System Settings*  *Target Host* , enter the host of the server (where the report is located) with a fully qualified domain, for example, cloud.saplumira.com.
5. For the *Service No.*, enter the port of the server.
6. On tab  *Login & Security*  *Security Options*  *SSL* , choose *Active*.
7. Save the configuration entries.

Configuring the Report Launchpad

To enable the launch of an SAP Lumira report from the SAP Hybris Marketing report launchpad in the UI, perform the following steps:

1. Logon to your AS ABAP system, and start the *Launchpad customizing* (transaction LPD_CUST).
2. Select *Role* HPA, *Instance* CUAN, and choose *Change*.
3. Choose *New Application*. Under  *Link Details*  *Link Text* , enter a name for the application launch.
4. For the *Application Type*, select *URL*.
5. Under  *Application Parameter*  *URL* , choose *URL Editor*. Enter the OpenDocument URL of the report (for which you want to enable the launch).
6. Under  *System*  *System Alias* , enter **LUMIRA_CLOUD**.

-
7. Choose *Show Advanced (Optional) Parameters*. Under **Application-Related Parameters** **Application Alias**, choose *Editor Application Alias*. Enter an application alias. You require the application alias when you create, or change the PFCG role that enables the report launchpad for the user.
 8. Under **Application-Related Parameters** **Additional Information**, enter **BOE**.
 9. Save the launchpad entry.

Creating or Changing a PFCG Role

To enable the report launchpad for a user in the SAP Hybris Marketing UI, you create or change a PFCG role. For the details, see chapter **Additional Applications or Facets** **Including Additional Applications** **Creating the Required PFCG Role** of this guide.

4 SAP Hybris Marketing Segmentation

4.1 Segmentation

Data Source

The *Segmentation* includes a generic segmentation engine that can be used to segment any kind of objects. Typically, the data for the segmentation population is provided by SAP HANA information models. The following modeling hints and customizing steps therefore focus on SAP HANA based data sources.

The *Segmentation* is shipped with standard SAP HANA information models based on replicated data from SAP ERP and/or SAP Customer Relationship Management (CRM). The *Segmentation* is preconfigured for the use of these standard models as a segmentation data source. If you want to adapt or extend the data source, or to use different data sources, take care of the following activities:

- Provide the SAP HANA information models for a custom data source.
- Adapt the standard customizing for the *Segmentation* to enable the use of the custom data source in the segmentation.

Preview Types

You can create custom visualizations for the attribute preview in the *Segmentation* and assign the preview types in Customizing for SAP Hybris Marketing. For the details, see section *Defining Additional Preview Types*.

4.1.1 Providing a Custom Data Source for the Segmentation

Prerequisites

You have a database user with development authorization in the SAP HANA studio. Use the modeler perspective to access the standard SAP HANA information models that are delivered with *Segmentation*. For a summary of the standard SAP HANA information models, see *Business Content* at <http://help.sap.com/mkt>.

i Note

The performance of the application depends on the performance of the SAP HANA information models and SQLscript procedures you provide. Familiarize yourself with the modeling guidelines to ensure that your information models and procedures are as performant as possible.

For more information about the modeling, see the *SAP HANA Modeling Guide*, and the *SAP HANA Developer Guide* at http://help.sap.com/hana_platform. In addition, consider the *SAP HANA Performance Analysis Guide* for information about how to identify and resolve performance issues of your SAP HANA database.

i Note

Make sure you use the correct format when you indicate the path and the name of SAP HANA information models, or SQLscript procedures:

- Lower case for the path, for example, `sap.hana-app.cuan.cpred.datafoundation`
- Upper case for the name of the model, or procedure, for example, `CA_CPRED_CRM_DS_PROD_AFFIN`
- Separate the path from the name by a slash `</>`.

Procedure

Adapting Standard SAP HANA Information Models

You can use copies of the standard SAP HANA information models that are shipped with *SAP hybris Marketing* for Segmentation and adapt them to your requirements. In the SAP HANA studio, locate the standard models and copy them to your custom package.

Creating Custom SAP HANA Information Models

As an alternative to the adaptation of copied standard models, you can create custom models and use it in Segmentation. Consider the following recommendations when creating custom models:

- Provide a proper description for each information model you create or change. The description is used as a title of the attribute grouping in the filter list of the Segmentation UI. In addition, consider the number of models you use for a set attributes since each model results in a separate attribute grouping. In the label column, provide descriptions for all fields to be displayed on the UI (with that description).
- Avoid to use mandatory variables, and mandatory input parameters.
- Use the same data type for key fields in all SAP HANA information models you assign (in Customizing) to a specific segmentation object.
- Consider the text joins that are required for fields with language dependent descriptions. In addition, complete the text mapping by renaming the label of the field description. For example, you have joined a table for the language dependent descriptions of `COUNTRY`. In the column `COUNTRY_ID`, rename the existing entry, such as `COUNTRY_TEXT` by `COUNTRY_ID.description`, and press Return.
- For analytic views or calculation views, make sure the model property *Multi Dimensional Reporting* is enabled. The setting is required for the provision of proper field descriptions in Segmentation. For the setting, the information model must have at least one key figure. If required, you can create a dummy key figure, make the setting, and delete the dummy key figure.
- For the client mapping, consider the following requirements:
 - The SAP HANA information models are defined as cross client.
 - The client for a particular schema, such as `SAP_CUAN_CRM` is retrieved as a derived input parameter from the client mapping table `CUANC_CLNTMAP`.
 - The client for the `ABAP/SAP_CUAN` schema is retrieved from the session context (`$$client$$`).
 - The data foundation for attribute views, analytic views, and graphical calculation views is filtered on the corresponding client.

4.1.2 Adapting the Standard Customizing for Segmentation

In the [Segmentation](#), the data source is referenced by a segmentation profile. [Segmentation](#) is shipped with standard segmentation profiles. The customizing activities described below allow for defining custom segmentation profiles.

Proceed with the following steps to adapt the preconfigured [Segmentation](#) settings for the use of custom data sources:

1. In the system, choose ► [Customizing - Edit Project \(transaction SPRO\)](#) ► [SAP Reference IMG](#) ► [SAP hybris Marketing](#) ► [Segmentation](#) ►

Note

For detailed information about each customizing activity, see the help topics that are assigned to the activities. In addition, use the F1 help for information about the customizing details.

2. Choose the customizing activity [Define Aliases for SAP HANA Data Sources](#). In this activity you provide an alias for each of your custom SAP HANA information model(s). In the following customizing activities, the aliases are used as a reference to the model(s) and thereby to the data source(s) for the Segmentation application.
As for all other [Segmentation](#) customizing activities, choose [New Entries](#) to create an item. Enter an arbitrary alias. For the data source, enter the location and the name of the model. Use SAP HANA Studio to gather the package and the technical name from the model properties. Enter the package plus model name in the following format: <package>/<name>. For example, `sap.hana-app.cuan.cseg/AT_CSEG_ERP_CUST_EXT`.
3. Choose the customizing activity [Define Segmentation Objects](#). With the segmentation objects you define in this activity you prepare for the assignment of data sources, which is done in the following customizing activity. Provide an arbitrary (technical) name and a business description for each segmentation object. When you set up segmentation objects, you can assign application related business logic by indicating a consumer class per segmentation object. See the help topic for more information about how to implement such business logic.
4. Choose the customizing activity [Assign SAP HANA Data Source to Segmentation Objects](#). In this activity, you use data source alias(es) to assign data source(s) to segmentation object(s).

Note

You can assign multiple data sources (using the alias) to one segmentation object.

In the customizing activity, click on the icon in the **attributes** column to see the list of attributes that are provided by the data source you have assigned to a segmentation object. For each attribute, a set of options is available. For the details, see the according F1 help. In addition, consider the following:

- For the visibility settings check how an attribute is to be used in the application, as a characteristic, or as a key figure.

If you use an attribute as a key figure:

- Consider that key figure dimensions allow for restricting key figure values to certain characteristics. Hence, you are able to create a restricted key figure by means of dimensions.
- Check the threshold and aggregation types that apply for the key figure.

If you use an attribute as a characteristic,

- Check whether it requires an additional semantic classification. Typically, date fields are affected, where the date format is missing (after data replication from the source system).

In the customizing activity, click on the icon in the **input parameters** column to see and edit the list of input parameter details. For the details, see the according F1 help.

Prerequisite: To enable the usage of input parameters when defining segmentation filters based on key figure attributes, define the input parameters in the SAP HANA information models (data source).

5. Choose the customizing activity [Define Segmentation Object Key Fields](#). In this activity you define key fields that allow to identify a segmentation object. For example, you define the key field **customer_number** for the segmentation object **ERP_customer**. In the next customizing activity, you assign an attribute to the key field.
6. Choose the customizing activity [Assign Segmentation Attributes to Object Key Fields](#). In this activity you assign a characteristic or key figure (attributes) to each segmentation object key field you have defined before. For example, assign the attribute **KUNNR** to the segmentation object key field **customer_number**.

Note

You can assign multiple attributes with the same business meaning but coming from different data sources. However, the attributes must share the same data type. For example, you can assign the attribute **KUNNR** and the attribute **ERPCUTSNO** (coming from different data sources) as they share the same data type **NVARCHAR(10)**.

7. Choose the customizing activity [Define Segmentation Profiles](#). In this activity, you determine combinations of data sources and segmentation objects. Each combination is represented by a segmentation profile. Each segmentation application references a segmentation profile, and thereby it accesses the data source you have defined. As a result, the segmentation profile determines the base population (population of the first segment) for a specific segmentation application.

Note that you can determine a default segmentation profile that is used when you start the segmentation application.

Note

You can assign multiple data sources and segmentation objects to one segmentation profile.

4.1.3 Disabling Segment Changes for which a Target Group has been Created

If you require additional logic for disabling segment changes, you can implement your own consumer class inherited from the standard consumer class as described in the extensibility guide under 7.2.3 (Adapting the Standard Customizing for [Segmentation](#), step 3).

You can use the BAdI *Disable Change/Remove of Segment Based on Assigned Target Group* (**BADI_CUAN_TG_DISABLE_SEG**) in the enhancement spot **CUAN_TARGET_GROUP** to implement your own consumer class. For more information, see the BAdI documentation.

4.1.4 Defining Additional Preview Types

You can implement and integrate additional visualizations for the characteristics and segment preview in the segmentation UI.

There are standard preview types in the [Segmentation](#), such as a pie chart for characteristics with a few values, a bar chart for characteristics with more values, and a map based visualization for the geographical characteristic [Country](#). In addition, the standard delivery includes a histogram preview for key figures.

i Note

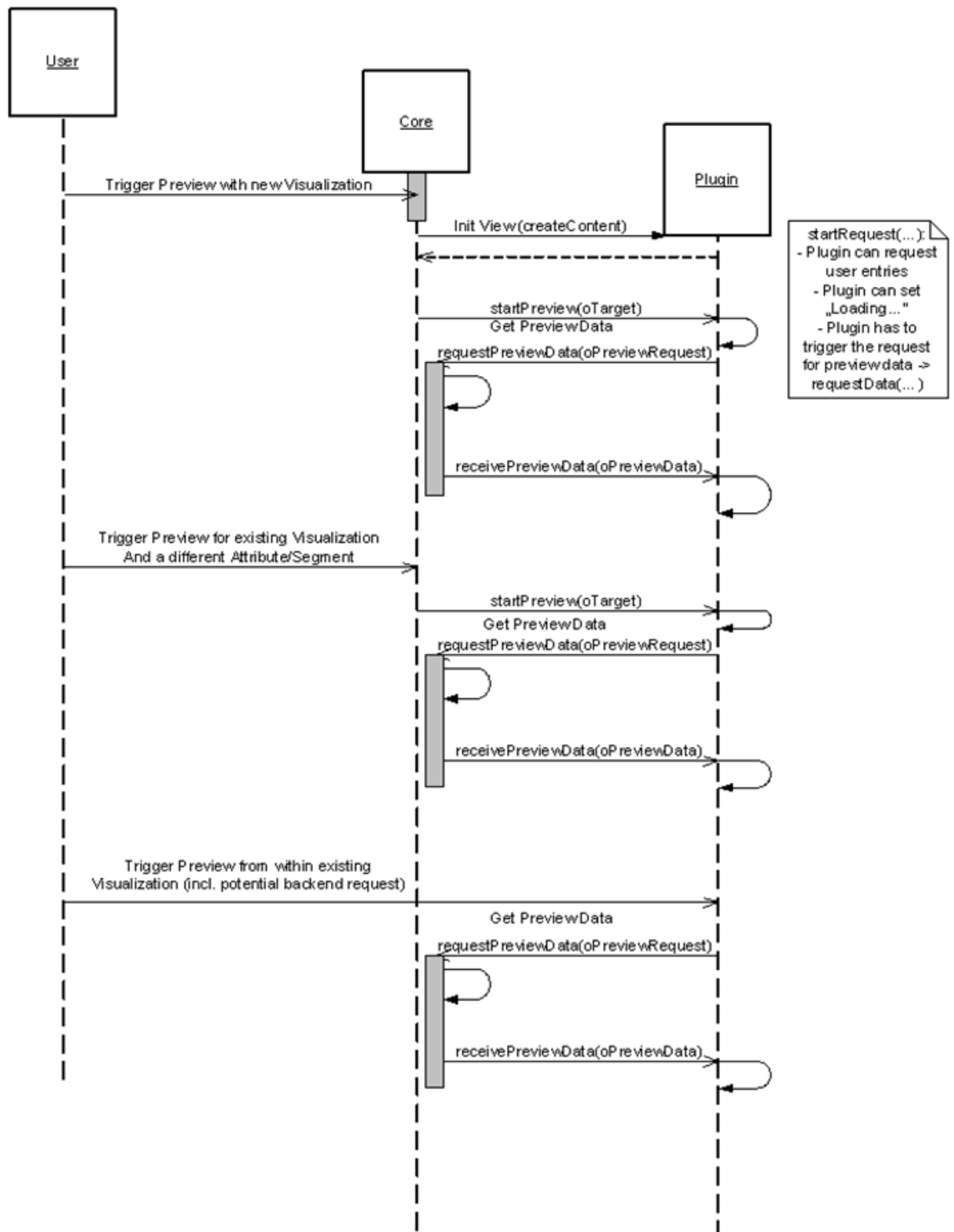
In the [Segmentation](#), the implementation of custom preview types is only supported for characteristics and segments, but not for key figures.

For more information about the standard preview types, see the Customizing under ► [SAP hybris Marketing](#) ► [Segmentation](#) ► [Define Segmentation Preview Types](#) ►.

You create a custom preview type implementing an SAPUI5 JavaScript view including a controller. For more information, see the Developer's Guide at <http://help.sap.com/nw-uiaddon> ► [Development Information](#) ►.

Interface

See the following chart for an overview over the interaction between the core segmentation UI and the SAPUI5 JavaScript view (plug-in):



When you create custom preview types based on SAPUI5 JavaScript you also need to implement a number of functions as explained in the following sections. There are functions in the plug-in that are called by the core UI of the segmentation, and there are functions in the core UI that can be called by the plug-in. For example, you request the data for the preview from the core UI calling the function `requestPreviewData()`.

Functions Provided by the Plug-in

In the custom preview type, you implement the following functions that are called by the segmentation core UI:

- `startPreview`

This function is called when the user starts a preview.

When implementing the function take care of the following:

- Provide a busy indicator that does not block the UI during the data load for the preview.
- Request the data for the preview, for example, by calling function `requestPreviewData()` for characteristics previews, or by calling your own data provider for segment previews,

The function has one parameter (a JavaScript object) that provides the following properties:

- Technical name of the attribute (`attName`)
- Language dependent text of the attribute (`attNameText`)
- Type of the attribute (`attType`), **characteristic** or **keyFigure**
- Data type (`dataType`), object that has the properties **decimals**, **length**, **type**
- ID of the preview visualization type as defined in customizing (`previewVis`)
- ID of the segment (`segmentID`)
- Language dependent description of the segmentation object (`segmentationObject`), for example, SAP ERP Customer
- Default currency code (`defaultCurrencyCode`)
- Array of input parameters filtered by the current `dataSourceAlias` (`aInputParams`). The objects of the array have the following properties:
 - `dataSourceAlias`
 - `dataSourceAliasDescription`
 - `dataType`
 - `paramName`
 - `text`
 - `defaultValue`
 - `mandatory`

- `receivePreviewData`

Implement the function only when you call the function `requestPreviewData()`. The function has one parameter (a JavaScript object) that provides the following properties:

- `attName`, for example, **COUNTRY**
- `attNameText`, for example, **Country**
- `dataComplete`, for example, **false**, which indicates that not all values from data base are part of the distribution array
- `dataSourceAlias`, for example, **SAP_ERP_CUSTOMER**
- `distribution`. The array contains the actual preview data. The objects of the array have the following properties:
 - `attValue` (dynamic type, usually string, or number)

- attValueText (string)
- attValueCount (number)
- searchIndex (string)

For example: [{"attValue": "DE", "attValueText": "Germany", "attValueCount": 3713, "searchIndex": "DE Germany"}, {"attValue": "US", "attValueText": "United States", "attValueCount": 3492, "searchIndex": "US United States"}]

- limitCount, for example, **20**. Sets the limit for the number of values returned from database
- offset, for example, **0**. Enables paged access. The offset **0** refers to the first page. For example, with the limit count **20**, and offset **0** you receive the entries 1- 20. With the offset **3**, and limit count **50**, you receive the entries 51 – 200.
- previewVis, for example, **SAP_CH_GEOMAP**. Specifies the preview type name allowing to use the same view for slightly different visualizations, such as preselected chart types, by customizing one SAPUI5 JavaScript view with two different names.
- searchString, for example, **""**. Contains the search text that filters the values received from the back end.
- segmentID, for example, **gseg_seg_lbl_0**
- remainderCount, for example, **123**. The cumulated count of hits for values that are not included in the distribution array when the count limit (limitCount) is exceeded.

If the function oPreviewData is set with an empty distribution array, it indicates that no data corresponds to the search criteria.

- **setError**

We recommend to implement the function `setError(sMessage)` that is called when a preview request leads to an error. Make sure that you stop the busy indicator in case of error.

Functions Provided by the Core UI

In the plug-in implementation for your custom preview type, you can use the following functions (provided by the segmentation core UI in the [ViewData](#) of your plug-in):

- **requestPreviewData()**

Use this function to start the data request for the characteristics preview with the following parameters:

- sSearchString. The search string for the filtering of the preview data. Use **""** for no filtering.
- iLimitCount. Specifies the maximum number of values returned from the back end.

- **requestAddSegment()**

Use this function to trigger the creation of a segment. The segment is created based on the attribute the user has selected. Use the following parameters to call the function:

- aFilterFromPreview. Set the array using one of the following alternatives:
 - With the selected attribute values in a flat array style, for example, **["DE", "CA", "US"]**
 - or
 - With objects that have the properties `valueLow` and `valueHigh`. Use the properties as filter-ranges with the select operator **between**. For example, **[{"valueLow": "0", "valueHigh": "10"}, {"valueLow": "20", "valueHigh": "50"}]**. Per range, you can additionally specify a mode for the interval (`intervalMode`). By default, the interval mode is **]]**; other modes are: **[[**, **[[**, or **]]**.
- iMode. This parameter specifies the segmentation mode as follows:
 - **0** (keep)

- 1 (separate)
- 2 (exclude)
- 3 (distribute)
- `aInputParams`. This optional array of input parameters (JavaScript objects) must have the following properties:
 - `dataSourceAlias`
 - `paramName`
 - `value`
- `getSegmentTreeJSON()`
Use this function to retrieve the JSON representation of the complete segmentation model tree.
- `applicationID`
Use this property to retrieve the application ID of the segmentation model tree (as a string), for example, `SAP_ADT`.
- `profileID`
Use this property to retrieve the profile ID of the segmentation model tree (as a string), for example, `SAP_ERP_CUSTOMER`.
- `segModelGUID`
Use this property to retrieve the GUID of the segmentation model tree (as a string).

4.1.5 Custom Fields in Segmentation

Business Context

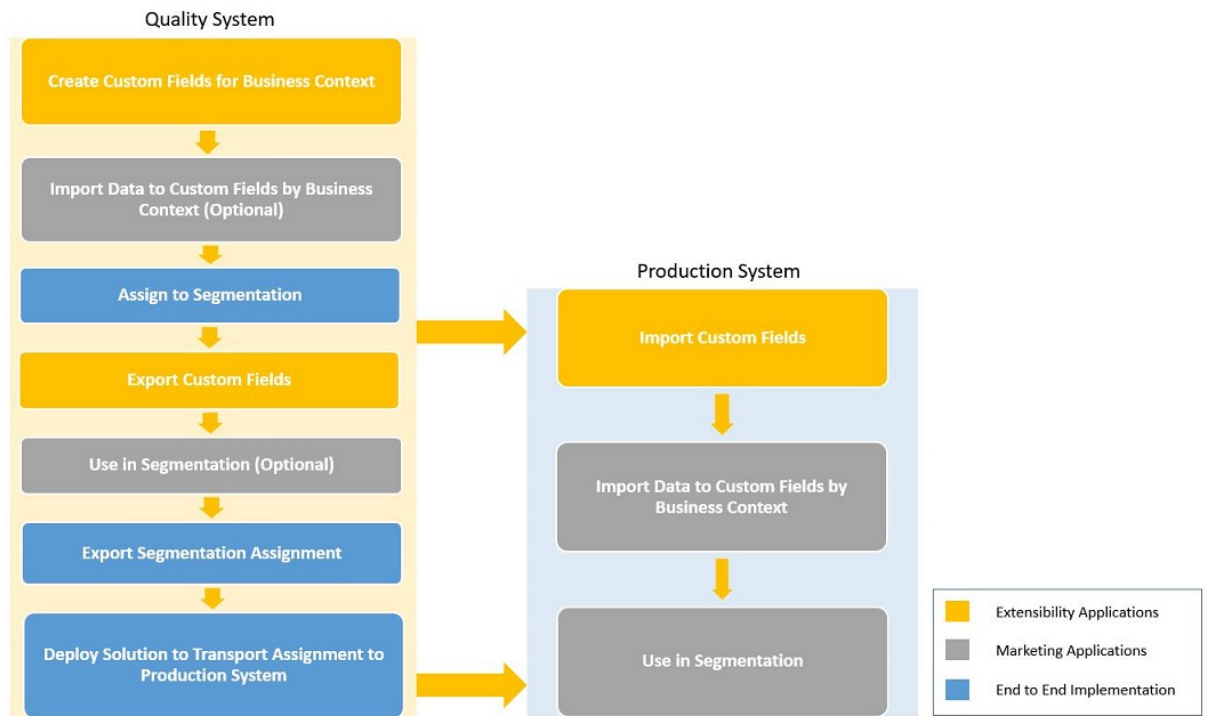
You want to assign a custom field to segmentation.

Prerequisites

- You have defined the custom fields based on your requirements. .
- You are assigned the following catalog roles:
 - `SAP_BCR_CA_IC_LND`
 - `SAP_BCR_CORE_EXT`
 - `SAP_BCR_CORE_SL_EXP`
 - `SAP_BCR_CEC_MKT_SEG_PC`

Settings

The following graphic illustrates the flow to create custom fields and use them in segmentation:



Follow the steps below to assign a custom field to segmentation:

1. Navigate to the [Add Custom Fields to Segmentation](#) app.
2. To assign custom fields of a business context to a segmentation profile, choose + (*Add*).
3. Select the required business context from the dropdown.
4. Select the segmentation object to which the custom fields are to be assigned.
5. In the *Custom Fields* section, check the fields and click *Set Visible* or *Set Invisible* to make them visible or invisible respectively.
6. Choose *Save*.
7. Export the segmentation assignment from the quality system to the production system.
8. Deploy the solution if you are in the Realize Phase or release your change if you are in the Change Phase to transport the assignment to the production system.

Note

After releasing your solution, wait until the transport reaches the production system.

9. Import the custom fields into your production system.
10. Use in segmentation.

4.1.6 Setting up the Geospatial Segmentation and Map Preview

This function allows you to view the distribution of geolocations on a map, for example, the distribution of customers in a region. You can then create new segments based on areas defined in the map.

Prerequisites:

Geocoordinates

To use geospatial segmentation, you need data in the form of geographic coordinates stored in your system. Addresses, for example, must first be converted to coordinates and then uploaded to the system. You can segment on different kinds of coordinates, such as the coordinates of interactions or shops in a certain area.

Access to Map Provider

For more information about the default configuration and how you can adapt it, see the [Installation and Configuration Guide](#) on <http://help.sap.com/mkt> ► [SAP Hybris Marketing On Premise](#) ► [SAP Hybris Marketing Product Page](#) ►.

To use the geospatial segmentation option with the map visualization, your browser requires access to the internet domain “here.com”. The domain provides the map data that is used in the geospatial segmentation option. Check, whether the users in question have access to this domain and consider the implications of communicating with servers outside your firewall.

Enabling a Geolocation Attribute

If you use a pair of numeric attributes to define the longitude and the latitude, set the [Semantic Type](#) of the longitude attribute in [Segmentation Configuration](#), ► [Segmentation Objects and Attributes](#) ► [Assigned Data Sources](#) ► [Geolocation Attribute](#) ► [Semantics](#) ►.

Use one of the following options:

- To process geolocations with coordinate (0,0) as actual geo points, set the longitude attribute to [Geo Point](#).
- To omit geolocations with coordinate (0,0) as initial values, set the longitude attribute to [Geo Point without \(0,0\)](#).

To complete the setting, define the latitude attribute as dependent on the longitude attribute. To do this, assign the latitude attribute under ► [Semantics](#) ► [Name of Dependent Attribute](#) ► to the longitude attribute.

i Note

For attributes of type `ST_POINT` or `ST_GEOMETRY`, the semantic type is automatically set.

Preview Type Assignment

By default, the [Geospatial Map](#) preview (SAP_CH_GEOSPATIAL_MAP) is assigned to the [Geolocation](#) attribute of type GEO_POINT. Note that you can only assign attributes of this type to the [Geospatial Map](#) preview, and vice versa.

Using a Custom Attribute

If you want to use the geospatial segmentation based on custom business data, make sure that your attribute universe provides the required geolocation information.

Use one of the following two options to model the according SAP HANA database tables, and SAP HANA information models:

- Define a column of (SQL data) type **ST_POINT**, or **ST_GEOMETRY** in your SAP HANA database table. Expose the column to your SAP HANA information model.
- In your SAP HANA database table, provide the longitude and latitude in two different numerical columns as a geolocation tuple. Expose the columns to your SAP HANA information model.
In addition to the modeling, configure the semantic type of the geolocation attribute in [Segmentation Configuration](#).

4.1.7 Restricting Data and User Authorization for Segmentation Models

Use Case

For marketing purposes, you want a segmentation model to only show customer data connected to a region A. Also, you want to make this segmentation model available only to marketers working in that same region A.

You can achieve this by carrying out two steps:

Restricting Data

You use segmentation profile filters to restrict the data available in the segmentation model.

1. Go to ► [Customizing](#) ► [Segmentation](#) ► [Define Segmentation Profiles](#) ► and click the icon in the filter column for the segmentation profile you want to filter.
2. In the table, enter the name of the filter attribute and specify the value.
In the example, the filter attribute would be [Region](#) with the specified value A.

Note

For detailed information about the customizing activity [Define Segmentation Profiles](#), see the assigned help topic. In addition, use the F1 help for information about the customizing details.

Restricting User Authorization

You use business roles to restrict user access (at the start of the generic segmentation application) to the segmentation model.

1. Go to [Role Maintenance](#) (transaction PFCG).
2. Assign a role to all the users you want to have restricted access to segmentation models only.
In the use case, these are the marketers in region A.
3. Now make custom entries for the role.
For the standard authorization object GSEG_START, enter the ID of the segmentation profile (GSEG_PROF) for that you want to grant the users access.
In the use case, this is the segmentation profile that has its available data restricted to region A.

Note

For more information on role maintenance, see the application documentation.

For more information on business roles and authorization objects, see the [Security Guide](#) for SAP Hybris Marketing on the SAP Help Portal at help.sap.com/mkt.

Alternatively, you can use the [BAI Segmentation Profile Authorization Check](#) to restrict user access.

Go to [Customizing](#) > [Segmentation](#) > [Business Add-Ins](#).

For detailed information, see the BAI documentation.

4.2 Target Groups

In SAP Hybris Marketing you can configure the member lists of target groups and enhance the target group details. The following chapter provides information on how to extend your system with these features.

4.2.1 Customer-Specific Member Lists

If you need to use a target group member list which differs from the standard member list, or requires a different data source because a segmentation object other than standard segmentation objects (SAP ERP Customer, SAP CRM Business Partner, SAP yMKT Interaction Contact) is used, the following options are available:

Caution

Please note that from 1602 you can make the required settings (except [Enabling Target Group Member List With Data From an External Source and Your Own Member Type](#)) in Customizing.

For more information, see the following chapter [Configure Member List in Customizing](#).

4.2.1.1 Configure Member List in Customizing

If you require adjustments to columns and, or data displayed in the target group member list, you can now configure the member list in Customizing.

For the Configurable Member List, the segmentation infrastructure is partially reused. Therefore you need to consider these interdependencies:

1. Whenever a target group is created in the system, it gets a segmentation object assigned.
 - Target groups created from within segmentation derive the segmentation object from the segmentation profile
 - Target groups created from outside segmentation follow the SAP-delivered defaults. You can overwrite these defaults in the customizing activity ► [Segmentation](#) ► [Target Group](#) ► [Set Default Segmentation Object per Member Type](#) ►.
2. The target group member list can be configured separately for each segmentation object.
3. All data you want to display in the target group member list needs to be in one single SAP HANA information model.

In addition, this SAP HANA information model needs to expose the key field(s) of the segmentation object for which you want to configure the member list.

- The SAP HANA information model needs to be maintained as data source in Customizing ► [Define Aliases for SAP HANA Data Sources](#) ►.
- The data source needs to be assigned to the required segmentation object in customizing ► [Segmentation](#) ► [Target Group](#) ► [Assign SAP HANA Data Sources to Segmentation Objects](#) ►. If you do not need or want to use attributes from this data source in Segmentation itself, unset the visibility of all attributes of this data source.
- The data sources columns which reflect the segmentation object keys need a mapping to the segmentation object keys in customizing ► [Segmentation](#) ► [Target Group](#) ► [Assign Segmentation Attributes to Object Key Fields](#) ►.

Now you can activate the Configurable Member List in Customizing:

To do this, go to ► [Segmentation](#) ► [Target Group](#) ► [Activate Features for Target Groups per Segmentation Object](#) ►. Then make your required settings in Customizing ► [Segmentation](#) ► [Target Group](#) ► [Configure Member List](#) ►.

In the list of attributes, you can use the following options:

- Set an attribute as visible.
- Define the position of an attribute.
- Define a navigation URL for an attribute.

Note

The placeholder in a URL is replaced with the actual value of the attribute when the link is navigated. For attributes with binary data types like GUIDs, the default binary encoding of OData is used, which is Base64.

If you want to have the attribute inserted into the URL as a normal string, set the [Semantic Type](#) of the attribute to [Globally Unique Identifier](#) in Customizing ► [Segmentation](#) ► [Assign SAP HANA Data Sources to Segmentation Objects](#) ► [Attributes](#) ►.

Excluding Attributes with Multiple Values

Caution

An attribute can have one value per target group member only. For example, the attribute [Age](#) has exactly one value per contact.

Attributes which have multiple values per target group member are not supported in the configurable member list for target groups.

Background: If an attribute has multiple values per target group member, the system displays more than one entry row per member in the target group member list. For example, the attribute [Interest](#) might have more than one value per contact. If the attribute [Interest](#) is added to the target group member list, the system displays one row for every hobby of every contact in the member list.

For more information on how you can configure your member list in Customizing, see the help topics assigned to the different activities and the F1 help.

Note

Please note that cases requiring your own target group member type are currently not possible in Customizing.

4.2.1.2 Enabling Target Group Member List With Data From an External Source

Caution

Please note that from 1602 you can make the required settings in Customizing.

For more information, see [Configure Member List in Customizing](#)[Configure Member List in Customizing \[page 93\]](#).

Use

If you need to use a target group member list which differs from the standard member list, or requires a different data source because a segmentation object other than standard segmentation objects (SAP ERP Customer, SAP CRM Business Partner, SAP CEI Interaction Contact) is used, follow the steps below.

A fixed number of fields is available for each member type provided in the standard.

If you want to use the same fields as in the standard, but want to retrieve data from different data source tables, or if you want to add additional attributes, you need to exchange the underlying attribute views. The following attribute views are used for the different member types, which are currently supported in the standard:

Table 5:

Member Type	Attribute View
01 (SAP ERP Customers)	sap.hana-app.cuan.common.internal/ AT_SEARCH_TG_MEMBER_CUST
02 (SAP CRM Business Partners)	sap.hana-app.cuan.common.crm.internal/ AT_SEARCH_TG_MEMBER_CUST_CRM
03 (SAP Customer Engagement Intelligence Interaction Contact)	sap.hana-app.cuan.common.internal/ AT_SEARCH_TG_MEMBER_IC

You need to copy the attribute views to a mapped package, enabling all standard queries to use these attribute views. To do so, proceed as follows:

- In SAP HANA Studio, go to the [Quick Launch](#) page and navigate to [Mass Copy](#).
- Configure a package mapping for the source package 'sap' to a customer package.
For details, refer to section [Copying Content Delivered by SAP](#) in the SAP HANA Modeling Guide on SAP Help Portal at <http://help.sap.com/hana>.
- Copy the respective attribute view and carry out your modifications to the copied attribute view (for example, by exchanging tables).

Note

The output structure of your new view must be the same as that of the original view. If you cannot fill some fields, you still have to add these columns to the output structure (for example, as calculated columns).

Note

If you are using member type 02 (CRM Business Partner), a time dependency for the address data is implemented in the standard query. Therefore the (calculated) attributes VALID_FROM and VALID_TO must be available and filled respectively.

If there is only a non-time dependent address record available, these attributes can be filled with default values 01.01.0001 and 31.12.9999 with the following formula:

VALID_FROM: string ('00010101000000')

VALID_TO: string ('99991231235959')

As the attributes, for example, PARENT_KEY, DB_KEY and DELETION_TYPE are required, you also have to include the member table CUAN_D_TG_MEMBER into your attribute view and join this table via attribute KUNNR (for member type 01 and 02) or MEMBER_KEY (for member type 03).

If you are using multiple clients, an additional join on attribute MANDT is required.

Afterwards, you need to clear the BOPF metadata buffer, so that the standard query automatically uses the copied customer-specific attribute view.

You do this in Customizing for SAP Hybris Marketing (formerly SAP Customer Engagement Intelligence), by choosing ► [General Settings](#) ► [Extensibility](#) ► [Clear BOPF Metadata from Buffer](#) ►.

Note

If you just want to add additional attributes, you do not need to carry out the steps in 4.3, but only proceed as stated above, and, in addition, as follows:

Extending the Include Structure

- In the back-end system, start the ABAP Dictionary (transaction SE11). Select Data type. Depending on your persistence requirements, enter one of the following include structures:
INCL_EEW_CUAN_TG_MEMBER if you want to persist the attribute
INCL_EEW_CUAN_TG_MEMBER_TR if you do not want to persist the attribute
- Choose [Change](#).
- In the [Change Structure](#) dialog, choose [Append Structure](#). Enter a name for the append structure.
- In the [Change Structure](#) dialog of the newly created append structure, enter the attribute(s) you want to add. Specify the attribute properties as required.
- Save and activate the append structure.

Caution

If you are using multiple member types, all underlying SAP HANA information models have to be enhanced accordingly.

4.2.1.3 Enabling Target Group Member List With Data From an External Source and Your Own Member Type

Use

If you need to use a target group member list which differs from the standard member list, or requires a different data source because a segmentation object other than standard segmentation objects (SAP ERP Customer, SAP CRM Business Partner, SAP CEI Interaction Contact) is used, follow the steps below.

Example 1: Using a Data Type Other Than the Standard Data Type

You want to use a customer-specific database table with different or multiple key fields as a basis for Segmentation.

Example 2: Using a Parallel Business

You want to use multiple data sources including your own member type, in parallel for Segmentation .

If you are using a parallel business, you need two different member types.

1. Creating your own member type

- Call up transaction SE11 and append the value with namespace 9*, x*, y*, or z* to the domain CUAN_TG_MEMBER_TYPE, via a fixed value append (in the context menu: ► [GoTo](#) ► [Fixed Value Append](#) ►).

- Redefine the method GET_MY_TYPE of class CL_CUAN_CUSTOMER_SEGMENTATION within the child class which is assigned in the segmentation customizing in the node "Define Segmentation Objects".
- To map the key of the segmentation object to the component in which the key value is stored in the target group member table, go to the [Segmentation Configuration](#) app and choose the activity [Mapping of Segmentation Object Keys to Member Components](#). For more information, see the corresponding chapter in the application help at <https://help.sap.com/> ► [SAP Hybris Marketing On Premise](#) ► [SAP Hybris Marketing Product Page](#) ► [Contacts and Profiles](#) ► [Segmentation](#) ► [Target Groups](#) ► [Target Group Configuration](#) ►.

2. Extending the Include Structure

If you want to display additional attributes, you have to extend the relevant include structure as follows:

- In the backend system, start the ABAP Dictionary (transaction SE11). Select [Data Type](#). Depending on your persistence requirements enter one of the following include structures:
INCL_EEW_CUAN_TG_MEMBER if you want to persist the attribute
INCL_EEW_CUAN_TG_MEMBER_TR or INCL_EEW_CUAN_TG_MEMBER_CALC if you do not want to persist the attribute

Note

If you enhance INCL_EEW_CUAN_TG_MEMBER_CALC, only the OData structure is enhanced, and not the structures for the standard queries.

- Choose [Change](#).
- In the [Change Structure](#) dialog, choose [Append Structure](#). Enter a name for the append structure.
- In the [Change Structure](#) dialog of the newly created append structure, enter the attribute(s) you want to add. Specify the attribute properties as required.
- Save and activate the append structure.

Caution

If you are using multiple member types, all underlying SAP HANA Information Models have to be enhanced accordingly.

3. Creating Customer-Specific SAP HANA Information Models for Target Group Members

- In the SAP HANA studio, you can either create your own SAP HANA information model(s), or copy and adapt the existing standard SAP HANA attribute view at `sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_CUST`.
- Ensure that the following mandatory fields are included in your SAP HANA information model(s):

Table 6:

Field	Table	Remark
DB_KEY	CUAN_D_TG_MEMBER	
PARENT_KEY	CUAN_D_TG_MEMBER	
DELETION_TYPE	CUAN_D_TG_MEMBER	
VERSION	CUAN_D_TG_MEMBER	

Field	Table	Remark
DELETION_VERSION	CUAN_D_TG_MEMBER	
KUNNR	CUAN_D_TG_MEMBER	
COUNTRY	Customer-specific	Required when using account authorization
NAME1	Customer-specific	Required if sorting is required

Caution

Ensure that the field names you use in your own model are identical to those in the standard SAP HANA information model, such as KUNNR for the account ID or COUNTRY.description for the country text field.

The field names are listed below under [Creating a Customer-Specific Result Structure](#).

Note

To search in the member list, you need to set the search property [Freestyle Search](#) to [True](#) for all attributes in which you want to carry out a fuzzy search.

4. Creating a Customer-Specific Result Structure

As you have to create a customer-specific query, you must create the corresponding result structure first. To do so, proceed as follows:

- Call up transaction SE11 and create a data structure in the customer namespace.
- Enter any required name in the [Short Description](#) field and enter the required fields under the [Components](#) tab.

Note

You can use any fields from the list below. However, you must ensure that the following mandatory fields are included in your result structure:

Table 7:

Field	Type	Description	Column in Attribute View	Mandatory
PARENT_KEY	/BOBF/CONF_KEY	Parent Key	PARENT_KEY	Yes
DB_KEY	/BOBF/CONF_KEY	Key	DB_KEY	Yes
DELETION_TYPE	CUAN_TG_DEL	Deletion Type	DELETION_TYPE	Yes
VERSION	CUAN_TG_VERSION	Version	VERSION	Yes

Field	Type	Description	Column in Attribute View	Mandatory
DELETION_VERSION	CUAN_TG_DEL_VERSION	Deletion Version	DELETION_VERSION	Yes
NAME1	AD_NAME1	Name	NAME1	Required if sorting is required
NAME_TEXT	AD_NAME1	Name (First Name and Last Name)	NAME_TEXT	
COUNTRY	LAND1	Country	COUNTRY	Yes, but only required when using account authorization
COUNTRY_T	LANDX	Country Description	COUNTRY.description	
REGION	REGIO	Region	REGION	
REGION_T	BEZEI20	Region Description	REGION.description	
CITY1	AD_CITY1	City	CITY1	
POST_CODE1	AD_PSTCD1	Postal Code	POST_CODE1	
STREET	AD_STREET	Street	STREET	
HOUSE_NUM1	AD_HSNM1	House Number	HOUSE_NUM1	
TELNR_LONG	AD_TELNRLG	Telephone Number	TELNR_LONG	
TELNR_MOBILE	AD_TELNRLG	Mobile Number	TELNR_MOBILE	
FAXNR_LONG	AD_FXNRLNG	Fax Number	FAXNR_LONG	
SMTP_ADDR	AD_SMTPADR	E-Mail Address	SMTP_ADDR	
PAFKT_T	VTEXT	Function	PAFKT.description	
PAFKT_FT	AD_FNCTN	Function Text		
ABTNR_T	VTEXT	Department	ABTNR.description	
ABTNR_FT	AD_DPRTMNT	Department Text		

- Create an appropriate table type for your result structure.

5. Enhancing a Business Object for Target Group Members

In order to create a customer-specific query, you must enhance the corresponding standard business object at which your customer-specific query is to be located. For example, this could be business object

CUAN_TARGET_GROUP, but also other business objects. If you want to use the account authorization, you have to enhance business object CUAN_CUSTOMER.

To enhance a business object, proceed as follows:

i Note

If you have already enhanced the business object due to a previous extension of SAP Hybris Marketing, you can skip this step.

- Call up transaction BOB.
- Choose the *Business Object Enhancement* pushbutton.
The *Create Enhancement* wizard appears.
- Choose *Continue* and enter the business object name (for example, CUAN_TARGET_GROUP) in the *Base Business Object* field.
- Enter a technical name in the customer namespace in the *Enhancement Name* field.
- Enter any required description in the *Description* field.
- You can leave the namespace field empty, but you have to specify a value (for example, Z) in the *Prefix* field.
- Choose *Continue*.
- Specify an entry in the *Constants Interface* field.
- Choose *Continue* and decide whether you are able to enhance your enhancement itself.
- Choose *Continue* and then *Complete*.

Your business object is now enhanced.

6. Creating a Customer-Specific Query

To create a customer-specific query, proceed as follows:

- Call up transaction BOB, select your enhanced business object from the previous step, and choose *Open* in the context menu.
- Select a node in the *Node Browser* section, for example, node CUSTOMER_MEMBER for business object CUAN_TARGET_GROUP, and choose *Create* in the context menu.
- Choose *Continue* in the *Create Query* wizard.
- Enter the required query name in the *Query Name* field. Enter any required query description in the *Description* field and choose *Continue*.
- Select the *Custom Query* radio button and choose *Continue*.
- Enter the following parameters for your customer-specific query:
Implementing Class: CL_CUAN_READ_TARGET_MEMBER
Data Type: CUAN_S_Q_TG_MEMBER_ELEMENTS
Result Type: Enter the name of the result type that you have created in section *Creating a Customer-Specific Result Structure*
Result Table Type: Enter the name of the table type that you have created in section *Creating a Customer-Specific Result Structure*
- Choose *Continue*.
- Enter the path to the corresponding SAP HANA information model that you created before, in the following format:
"_SYS_BIC"."<your_path_in_SAP_HANA_Modeler>/<name_of_your_information_model>"
Example: "_SYS_BIC"."sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_CUST"
- Choose *Complete* to finish creating your query.

Note

If you want to enhance the logic of your query (for example, by setting additional filters), you can create your own class with superclass `CL_CUAN_READ_TARGET_MEMBER`, and assign it as an implementing class to the query.

7. Adapting Customizing

- Assign your newly created query name with its business object name and node name to your newly created member type.

You do this in Customizing for SAP Hybris Marketing (formerly SAP Customer Engagement Intelligence), by choosing **Target Group** **Extensibility of Member List** **Define Query Access for Member Type**.

Note

If you want to replicate the target group to SAP CRM, the new member type must be considered in the implementation of `BAdI CUAN_CRM_REPL_TARGET_GROUP`.

8. Adding Your Own Fields to the Member List

If you are using your own member type, the facet of the member type 01 (SAP ERP Customers) is displayed as default.

You can add your own fields to the member list for a customer-specific member type, using the following controller hook:

Table 8:

Controller	Hook
<code>CUAN_TG_TI.cuan_tg_ti.TargetGroupMembersTBV</code>	<code>extHookGetMembersTable</code>

The controller hook returns an instance of `sap.ui.table.Table` or `sap.hpa.cuan.util.HPATable`. As an example implementation, you can use the function `getErpTable()` in `TargetGroupMembersTBV.view.js`.

Parameters passed are: `sMemberType` (member type of the target group) and `oErpTableTemplate`.

You can use table containing columns of the ERP accounts as a template. You can add new columns or remove existing columns instead of having to create a new table.

9. Hiding Standard Fields in the Member List

If you are using your own member type, the facet of the member type 01 (SAP ERP Customers) is displayed as default.

You can hide standard fields from the member list, using the following controller hook:

Table 9:

Controller	Hook
<code>CUAN_TG_TI.cuan_tg_ti.TargetGroupMembersTBV</code>	<code>extHookIsPropertyHidden</code>

The controller hook returns true if you want to hide standard fields.

Parameters passed are: `sMemberType` (member type of the target group) and `sPropertyName` (property name of the entity set `TargetGroupMember`).

4.2.1.4 Enabling Target Group Member List for a Standard Member Type With Attributes Other Than Those in the Standard Member Type

Caution

Please note that from 1602 you can make the required settings in Customizing.

For more information, see [Configure Member List in Customizing](#)[Configure Member List in Customizing \[page 93\]](#).

Use

If you need to use a target group member list which differs from the standard member list, or requires a different data source because a segmentation object other than standard segmentation objects (SAP ERP Customer, SAP CRM Business Partner, SAP CEI Interaction Contact) is used, follow the steps below.

You are using your own query for the standard member type, and you want to enhance the standard query logic, for example, by setting additional filters (which cannot be set in the SAP HANA information model), or by excluding attributes.

Note

If you only want to add new fields, see the steps described in section 4.1, [Enabling Target Group Member List With Data From an External Source](#).

To do so, proceed as follows:

1. Creating Customer-Specific SAP HANA Information Models for Target Group Members

- In the SAP HANA studio, you can either create your own SAP HANA information model(s), or copy and adapt the existing standard SAP HANA attribute views at `sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_CUST`, `sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_CUST_CRM` and `sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_IC`.
- Ensure that the following mandatory fields are included in your information model(s):

Table 10:

Field	Table	Remark
DB_KEY	CUAN_D_TG_MEMBER	
PARENT_KEY	CUAN_D_TG_MEMBER	
DELETION_TYPE	CUAN_D_TG_MEMBER	
VERSION	CUAN_D_TG_MEMBER	

Field	Table	Remark
DELETION_VERSION	CUAN_D_TG_MEMBER	
KUNNR	CUAN_D_TG_MEMBER	
COUNTRY	Customer-specific	Required when using account authorization
NAME1/NAME_TEXT	Customer-specific	Required if you are using the standard member list view with sorting (NAME1 is required for member type 01 and 02 and NAME_TEXT for member type 03)

Caution

Ensure that the field names you use in your own model are identical to those in the standard SAP HANA information model, such as KUNNR for the account ID or COUNTRY.description for the country text field.

The field names are listed below under [Creating a Customer-Specific Result Structure](#).

Note

To search in the member list, you need to set the search property *Freestyle Search* to *True* for all attributes in which you want to carry out a fuzzy search.

2. Creating a Customer-Specific Result Structure

As you have to create a customer-specific query, you must create the corresponding result structure first. To do so, proceed as follows:

- Call up transaction SE11 and create a data structure in the customer namespace.
- Enter any required name in the *Short Description* field and enter the required fields under the *Components* tab.

Note

You can use any fields from the list below. However, you must ensure that the following mandatory fields are included in your result structure:

Table 11:

Field	Type	Column in Attribute View	Mandatory
PARENT_KEY	/BOBF/CONF_KEY	PARENT_KEY	Yes
DB_KEY	/BOBF/CONF_KEY	DB_KEY	Yes
DELETION_TYPE	CUAN_TG_DEL	DELETION_TYPE	Yes
VERSION	CUAN_TG_VERSION	VERSION	Yes

Field	Type	Column in Attribute View	Mandatory
DELETION_VERSION	CUAN_TG_DEL_VERSION	DELETION_VERSION	Yes
KUNNR	KUNNR	KUNNR	Yes
MEMBER_KEY	/BOBF/CONF_KEY	MEMBER_KEY	Yes
NAME1	AD_NAME1	NAME1	Required for member type 01 and 02 if you are using the standard member list view with sorting
NAME_TEXT	AD_NAME1	NAME_TEXT	Required for member type 03 if you are using the standard member list view with sorting
COUNTRY	LAND1	COUNTRY	Required when using account authorization
COUNTRY_T	LANDX	COUNTRY.description	
REGION	REGIO	REGION	
REGION_T	BEZEI20	REGION.description	
CITY1	AD_CITY1	CITY1	
POST_CODE1	AD_PSTCD1	POST_CODE1	
STREET	AD_STREET	STREET	
HOUSE_NUM1	AD_HSNM1	HOUSE_NUM1	
TELNR_LONG	AD_TELNRLG	TELNR_LONG	
TELNR_MOBILE	AD_TELNRLG	TELNR_MOBILE	
FAXNR_LONG	AD_FXNRLNG	FAXNR_LONG	
SMTP_ADDR	AD_SMTPADR	SMTP_ADDR	
PAFKT_T	VTEXT	PAFKT.description	
PAFKT_FT	AD_FNCTN		
ABTNR_T	VTEXT	ABTNR.description	
ABTNR_FT	AD_DPRTMNT		

- Create an appropriate table type for your result structure.

3. Enhancing a Business Object for Target Group Members

In order to create a customer-specific query, you must enhance the corresponding standard business object at which your customer-specific query is to be located. For example, this could be business object CUAN_TARGET_GROUP, but also other business objects. If you want to use the account authorization, you have to enhance business object CUAN_CUSTOMER.

To enhance a business object, proceed as follows:

i Note

If you have already enhanced the business object due to a previous extension of SAP Hybris Marketing, you can skip this step.

- Call up transaction BOB.
- Choose the *Business Object Enhancement* pushbutton.
The *Create Enhancement* wizard appears.
- Choose *Continue* and enter the business object name (for example, CUAN_TARGET_GROUP) in the *Base Business Object* field.
- Enter a technical name in the customer namespace in the *Enhancement Name* field.
- Enter any required description in the *Description* field.
- You can leave the namespace field empty, but you have to specify a value (for example, Z) in the *Prefix* field.
- Choose *Continue*.
- Specify an entry in the *Constants Interface* field.
- Choose *Continue* and decide whether you are able to enhance your enhancement itself.
- Choose *Continue* and then *Complete*.

Your business object is now enhanced.

4. Creating a Customer-Specific Query

To create a customer-specific query, proceed as follows:

- Call up transaction BOB, select your enhanced business object from the previous step, and choose *Open* in the context menu.
- Select a node in the *Node Browser* section, for example, node CUSTOMER_MEMBER for business object CUAN_TARGET_GROUP, and choose *Create* in the context menu.
- Choose *Continue* in the *Create Query* wizard.
- Enter the required query name in the *Query Name* field. Enter any required query description in the *Description* field and choose *Continue*.
- Select the *Custom Query* radio button and choose *Continue*.
- Enter the following parameters for your customer-specific query:
Implementing Class: CL_CUAN_READ_TARGET_MEMBER
Data Type: CUAN_S_Q_TG_MEMBER_ELEMENTS
Result Type: Enter the name of the result type that you have created in section *Creating a Customer-Specific Result Structure*
Result Table Type: Enter the name of the table type that you have created in section *Creating a Customer-Specific Result Structure*
- Choose *Continue*.
- Enter the path to the corresponding SAP HANA information model that you created before, in the following format:
"_SYS_BIC". "<your_path_in_SAP_HANA_Modeler> / <name_of_your_information_model>"
Example: "_SYS_BIC"."sap.hana-app.cuan.common.internal/AT_SEARCH_TG_MEMBER_CUST"

- Choose [Complete](#) to finish creating your query.

Note

If you want to enhance the logic of your query (for example, by setting additional filters), you can create your own class with superclass `CL_CUAN_READ_TARGET_MEMBER` and assign it as an implementing class to the query.

5. **Adapting Customizing**

Assign your newly created query name with its business object name and node name to the relevant member type.

You do this in Customizing for SAP Hybris Marketing (formerly SAP Customer Engagement Intelligence), by choosing [Target Group](#) > [Extensibility of Member List](#) > [Define Query Access for Member Type](#).

6. **Hiding Standard Fields in the Member List**

If you are using your own member type, the facet of the member type 01 (SAP ERP Customers) is displayed as default.

You can hide standard fields from the member list, using the following controller hook:

Table 12:

Controller	Hook
<code>CUAN_TG_TI.cuan_tg_ti.TargetGroupMembersTBV</code>	<code>extHookIsPropertyHidden</code>

The controller hook returns true if you want to hide standard fields.

Parameters passed are: `sMemberType` (member type of the target group) and `sPropertyName` (property name of the entity set `TargetGroupMember`).

4.2.1.5 Enabling Target Group Member List for More Complex Scenarios

Caution

Please note that from 1602 you can make the required settings in Customizing.

For more information, see [Configure Member List in Customizing](#) [Configure Member List in Customizing](#) [page 93].

Use

If you need to use a target group member list which differs from the standard member list, or requires a different data source because a segmentation object other than standard segmentation objects (SAP ERP Customer, SAP CRM Business Partner, SAP CEI Interaction Contact) is used, follow the steps below.

For more complex scenarios, you can control the retrieval of target group member data by implementing Business Add-In (BAdI) CUAN_TG_MEMBER_RETRIEVE. For example, if user/role-specific attributes should be displayed, you can implement this logic in the BAdI.

For more information on how to implement the BAdI, see the documentation in SAP Hybris Marketing (formerly Customizing for SAP Engagement Intelligence), by choosing [Target Group](#) > [Extensibility of Member List](#) > [BAdI: Retrieval of Target Group Member Data with Own Query](#).

4.2.2 Enhancing the Target Group Details

You have the following options for enhancing the target group details:

4.2.2.1 Adding New Fields to Key Information

You can add your own fields to the target groups key information section using the following view and extension point:

Table 13:

View	Extension Point
CUAN_TG_TI.cuan_tg_ti.TargetGroupThingInspectorFragment	extensionKeyInformation

4.2.2.2 Adding Your Own Actions

You can add your own actions to replace the actions delivered with the standard, using the following view and extension point:

Table 14:

View	Extension Point
CUAN_TG_TI.cuan_tg_ti.TargetGroupThingInspectorFragment	extensionThingActions

4.2.2.3 Defining the Layout of the Business Card

You can define the layout of the business card used for target group members of member type [Interaction Contact](#), using the following controller and hook:

Table 15:

Controller	Hook
CUAN_TG_TI.cuan_tg_ti.TargetGroupMembersGRIDV	extHookIsConsumerLayoutType

4.2.2.4 Adding Your Own Initialization Steps

You can add your own initialization steps, for example, enhance the [i18n](#) resource model, using the following controller and hook:

Table 16:

Controller	Hook
cuan_tg_ti.TargetGroupThingInspector	extHookOnInit

4.2.3 Adding New Fields and Icon Tabs to the Release Target Groups SAP Fiori App

Use

You can extend [Release Target Groups](#) according to your business needs for different aspects. For this purpose, the following extensibility options are available:

- Add new fields to the existing icon tab [Key Information](#)
- Add a new icon tab to the target group details

To add new fields to an existing icon tab, or create a new icon tab, the following extensibility entities are available on the different software layers. You have to extend each of these entities according to your specific business needs:

Table 17:

UI		Backend/ABAP	
View	Extension Point	Design Time: Gateway Entity	Design Time: Extension Include (in DDIC Structure)
S3.view.app.xml	extensionKeyInformation	TargetGroup	INCL_EEW_CUAN_TG_ROOT

UI		Backend/ABAP	
View	Extension Point	Design Time: Gateway Entity	Design Time: Extension Include (in DDIC Structure)
S3.view.app.xml	extensionIconTabFilter	TargetGroup	INCL_EEW_CUAN_TG_ROOT

If you want to extend the target group header, you need to use the above extension include.

For information on how to extend the extension include, see chapter 1, [Extending the Data Source and Business Objects](#).

Business Add-Ins

The following BADIs are available for extensibility purposes:

- CUAN_TG_MEMBER_RETRIEVE (Retrieval of Target Group Member Data with own Query)
You can use this BAdI to implement a different method to retrieve data for the entity TargetGroupMember of OData service CUAN_COMMON_SRV. For example, you can replace the SAP Hybris Marketing query TARGET_MEMBER_BY_ELEMENTS with your own query, built on top of your own SAP HANA information model.
To view this BAdI, see Customizing for SAP Hybris Marketing ► [Target Group](#) ► [Retrieval of Target Group Member Data with own Query](#) ►.

More Information

For a general description of the extensibility options and procedures of SAP Fiori apps, see ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) ► in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori> ►.

For more information about extension includes, see ► [SAP Fiori for SAP Business Suite](#) ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) ► [Checking the SAP-Enabled Extension Options](#) ► [Extension Includes](#) ► in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori> ►.

5 SAP Hybris Marketing Recommendation

5.1 Recommendation

Data Source and Data Source Pre-Filter

SAP Hybris Marketing Recommendation includes a generic recommendation engine that accepts any kind of user, item, or transaction. Recommendation ships with standard SAP HANA information models that are based on replicated data from one or more of the following systems: SAP ERP, SAP Hybris Marketing Data Management, or SAP Hybris Marketing Segmentation. Recommendation is pre-configured for the use of these standard data sources but you can adapt or extend them, or use different data sources entirely.

You customize data sources as follows:

- Provide SAP HANA information model(s) for the custom data source
- Adapt the standard customizing for Recommendation to enable the use of the custom data source

Algorithms

Recommendation depends on a set of base algorithms such as APRIORI LITE, Structure Query Language (SQL) association, and Collaborative Filtering using Cosine Similarities. These base algorithms provide recommendations when they are fed with users, items, and transactions.

You adapt or extend base algorithms as follows:

- Implement a runtime SAP HANA procedure
- Implement a model generation and cleanup SAP HANA procedure (if needed)
- Adapt the standard customizing for Recommendation to enable the use of the custom algorithm

When you modify standard customizing, you must regenerate the stored procedure for Recommendation. You do this by executing the procedure in Customizing for [SAP Hybris Marketing](#) under ► [Recommendation](#) ► [Generate SAP HANA Stored Procedures](#) ►.

Rules

In addition to the standard algorithms, rule-based algorithms can be used by marketers for selecting, re-ranking, and filtering recommendations. Currently, two types of recommendations are supported: offers and products.

Rule-based algorithms are based on the SAP HANA rules framework and can be maintained in the rule editor. Rules can be combined by using different rule fragments, in the rule editor. A rule fragment is a pre-defined expression that is used to select, re-rank, or filter a data pool.

5.1.1 Providing a Custom Data Source or Data Source Pre-Filter

Use SAP HANA Studio to provide SAP HANA information models and SQLScript procedures to Recommendation. The system performance of Recommendation depends on the SAP HANA information models and SQLScript procedures that you provide. We therefore recommend that you familiarize yourself with modeling guidelines to ensure that your information models and procedures are optimized to perform well. For more information about modeling, see the SAP HANA Modeling Guide and the SAP HANA Developer Guide at http://help.sap.com/hana_platform. Furthermore, see the SAP HANA Performance Analysis Guide for information about how to identify and resolve performance issues in your SAP HANA database.

Providing a Custom Data Source

Create either an SAP HANA attribute view or an SAP HANA calculation view to expose the appropriate data according to the data source type (item, user, transaction).

Providing a Custom Item Data Source

Perform the following steps to provide a custom item data source:

1. Create an SAP HANA attribute view or an SAP HANA calculation view to expose the item ID and description. The view must pre-filter data for the specific customer and language.
2. In Recommendation, define a new data source class item type, and enter the following information:

Table 18:

Field	Required Value
Name	The name of the item data source. This name appears in the UI.
Data Source Class	Item
Artifact	SAP HANA view
Artifact Name	The name of the newly created SAP HANA view
Item Name Attribute	The name of the attribute that has the item ID
Item Description Attribute	The name of the attribute that has the item description
Item Data Source Type	Select the newly created data source type

Providing a Custom User Data Source

Perform the following steps to provide a custom user data source:

1. Create an SAP HANA attribute view or an SAP HANA calculation view to expose the user ID and full name. The view must pre-filter data for the specific customer.
2. In Recommendation, define a new data source class item type, and enter the following information:

Table 19:

Field	Required Value
Name	The name of the user data source. This name appears in the UI.
Data Source Class	User
Artifact	SAP HANA view
Artifact Name	The name of the newly created SAP HANA view
User Attribute	The name of the attribute that has the user ID
User Name Attribute	The name of the attribute that has the user full name
Item Data Source Type	Select the newly created data source type

Providing a Custom Transaction Data Source

Perform the following steps to provide a custom transaction data source:

1. Create an SAP HANA attribute view or an SAP HANA calculation view to expose the transaction ID, user ID, and item ID. The view must pre-filter data for the specific customer.
2. In Recommendation, define a new data source class item type, and enter the following information:

Table 20:

Field	Required Value
Name	The name of the transaction data source. This name appears in the UI.
Data Source Class	Transaction
Artifact	SAP HANA view
Artifact Name	The name of the newly created SAP HANA view
Transaction Attribute	The name of the attribute that has the transaction ID
Transaction Data Source Type	Select the newly created data source type
Item Name Attribute	The name of the attribute that has the item ID
Item Data Source Type	Select the data source corresponding to the item ID
User Attribute	The name of the attribute that has the user ID

Field	Required Value
User Data Source Type	Select the data source corresponding to the user ID

Providing a Custom Data Source Pre-Filter

Perform the following steps to provide a custom transaction data source:

1. If required, create an SAP HANA attribute view or an SAP HANA calculation view to expose the following:
 - Either the transaction ID (in the case of a transaction ID), user ID (in the case of a user pre-filter), or item ID (in the case of an item pre-filter)
 - The field that should be pre-filtered
2. In Recommendation, create a new data source pre-filter and enter the following information:

Table 21:

Field	Required Value
Name	Name of the pre-filter that displays in the UI
Data Source Type	Select the data source to which pre-filtering is applied
Param. Value Type	Select the type of data that can be entered
Value Help Type	Select the source of the data for the value help
Artifact Name	The SAP HANA view name where pre-filtering is defined
Artifact Type	SAP HANA View
Key Attribute	The fields representing the ID (either user, item, or transaction)
Pre-Filter Attribute	The field representing the pre-filter
Desc. Artifact Type	The HANA content type for the value help
Desc. Artifact Name	The name of the SAP HANA view that provides the value help ID and description
Key Attribute	The pre-filter ID attribute
Desc. Attribute	The pre-filter description attribute
Language Key	The language key attribute
Field Domain Name	The name of the ABAP Domain (if the value is coming from the domain)

3. Go to the data source and associate the newly created data source pre-filter to the data source.

5.1.2 Providing a Custom Algorithm

Recommendation is delivered with a set of recommendation algorithms. You can extend this set in [Customizing for SAP Hybris Marketing](#) under ► [Recommendation](#) ► [Algorithms Definition](#) ► [Define Algorithms](#) ►.

When extending the recommendation algorithms, provide the following:

- Runtime Stored Procedure
- Model Generation Stored Procedure (optional)
- Cleanup Stored Procedure (optional)

Runtime Store Procedure Interface

When creating a SAP HANA stored procedure, you must implement the following interface:

Table 22:

Input		
inResultSet Table		
ENGINE_ID	VARCHAR	30
RESULT_ITEM	VARCHAR	18
RESULT_ITEM_TYPE	VARCHAR	2
SCORE	INTEGER	
inTask Table		
TASK_ID	INTEGER	30
MAX_RESULT	INTEGER	18
ENGINE_ID	VARCHAR	30
inLeadingItem		
ENGINE_ID	VARCHAR	30
LEADING_ITEM	VARCHAR	18
LEADING_ITEM_TYPE	VARCHAR	2
inBasketItem		
ENGINE_ID	VARCHAR	30
ITEM_ID	VARCHAR	18

Input		
ITEM_TYPE	VARCHAR	2
inContext		
PREFILTER_PARAM_ID	VARCHAR	32
PARENT_PREFILTER_PARAM_ID	VARCHAR	32
DATASOURCE_PARAM_ID	VARCHAR	32
PARAMETER_VALUE	NVARCHAR	100
inTaskParam		
TASK_ID	INTEGER	
ALGO_TYPE_PARAM_ID	NVARCHAR	32
PARAM_OPTION	NVARCHAR	2
VALUE_FROM	NVARCHAR	100
VALUE_TO	NVARCHAR	100
inUserId	NVARCHAR	30
inUserId	NVARCHAR	2

Table 23:

Output		
outResultSet		
TASK_ID	VARCHAR	30
RESULT_ITEM	VARCHAR	18
RESULT_ITEM_TYPE	VARCHAR	2
SCORE	INTEGER	
outContext		
PREFILTER_PARAM_ID	VARCHAR	32
PARENT_PREFILTER_PARAM_ID	VARCHAR	32
DATASOURCE_PARAM_ID	VARCHAR	32
PARAMETER_VALUE	VARCHAR	100

Model Generation Stored Procedure Interface (Optional)

When creating an SAP HANA stored procedure, you must implement the following interface:

Table 24:

Input		
inTaskId	INTEGER	
inDocument		
TRANSACTION_ID	VARCHAR	10
ITEM_ID	VARCHAR	18
ITEM_TYPE	VARCHAR	2
USER_ID	VARCHAR	16
USER_TYPE	VARCHAR	2
inUser		
USER_ID	VARCHAR	16
USER_TYPE	VARCHAR	2
inItem		
ITEM_ID	VARCHAR	16
ITEM_TYPE	VARCHAR	2


Cleanup Stored Procedure Interface (Optional)

When creating an SAP HANA stored procedure, you must implement the following interface:

Table 25:

Input		
inEngineId	VARCHAR	30
inModelId	VARCHAR	32

5.1.3 Configuring Data Sources for Algorithms

The *Configuring Algorithms* Customizing for *SAP Hybris Marketing* under  *Recommendation* allows you to define data sources to feed particular algorithms. You can use the activity to configure new data sources for

existing algorithms, assign existing data sources to new algorithms, or create new data sources for new algorithms.

5.1.4 Configuring Rule-Based Recommendations

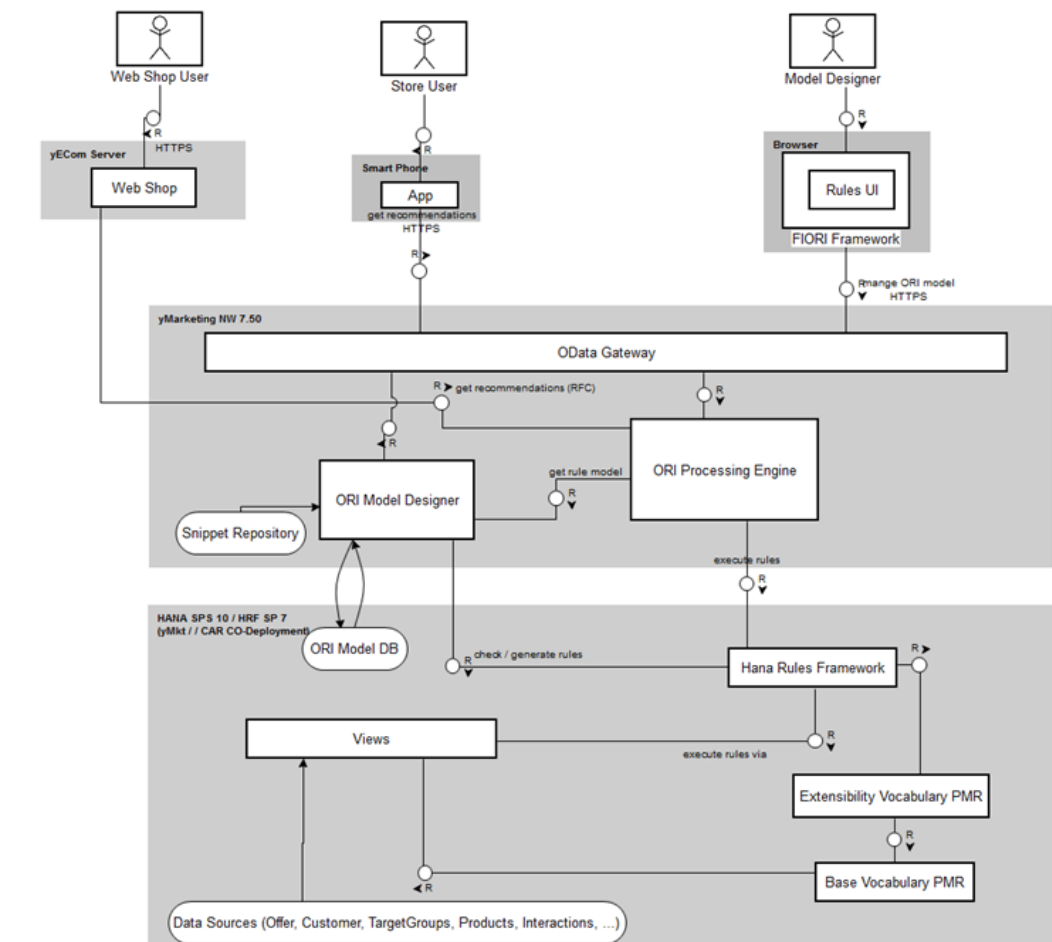
This document includes a technical overview of rule-based recommendations, and the different levels of enhancements.

Technical Overview

This section describes the technical components that are used for rule-based recommendations. As the business information on which the recommendation can be based is stored in SAP HANA database tables, this information must be accessed via SQL. On the model designer level, a meta language (rule fragments) is used to allow easy rule modelling by a non-technical user. In between model design and SAP HANA database, a transformation from meta language into executable SQL takes place.

Technical Components

The following diagram shows the technical components:



Important Changes in the Rule Fragment Repository

The rule fragments are no longer available in ABAP class `CL_PRECO_RUL_SNIPPET_UTIL`, they are now part of an HRF vocabulary (modeled as HRF aliases).

New Structure of the HRF Vocabularies

For the new structure of the recommendation-related HRF vocabularies, see section [New Rule Fragment in Existing Attributes](#) below.

1. With the rule UI at the level of model designer, existing rule fragments can form a business rule. The following example shows the rule fragment *With Product ID* from the *Snippet Repository* (snippet = technical name of rule fragment):

Example

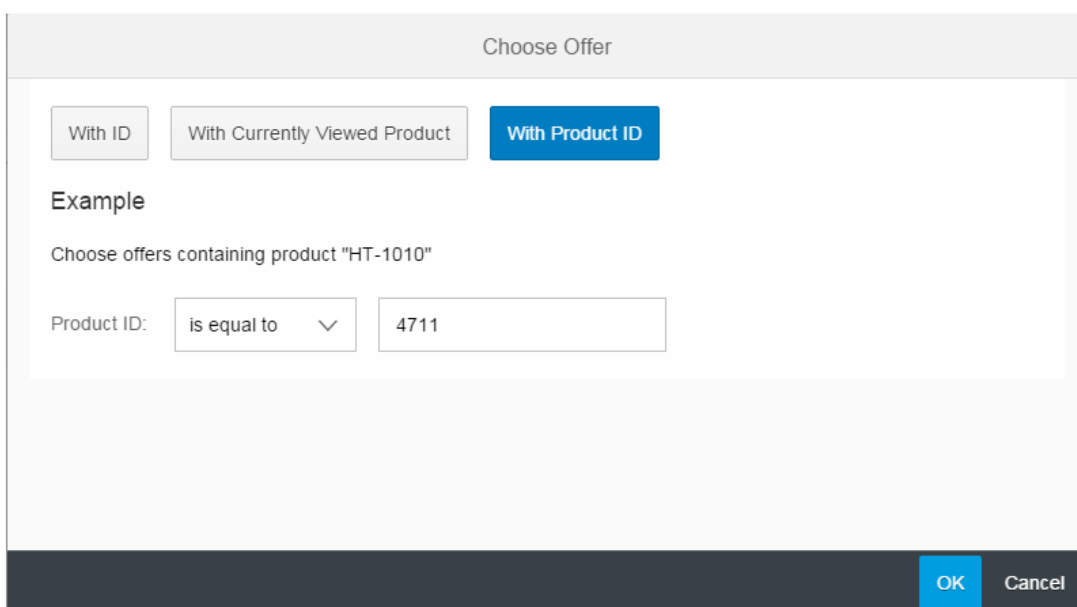




Table 26:

Technical Component	Description
HRF alias name (rule fragment ID)	s02_04
HRF alias content	<code>' ' exists in (Offer.Product.External_Id)</code>

Technical Component	Description
HRF alias externalMetadata (additional rule fragment metadata related to UI rendering)	<pre>{ "descriptionResKey": "PUSH_OFFER_WITH_PRODUCT", "reskeyParamToToken" : "0(len:18)", "relatedStepType": "O", "objectClass": "OFFER", "exampleResKey": "PUSH_OFFER_WITH_PRODUCT", "resultType": "12", "contextParams": [{ "id": "ofr_location", "resourceKey": "OFR_LOCATION" }, { "id": "ofr_date", "resourceKey": "OFR_DATE" }] }</pre>
descriptionResKey	Resource key suffix for the rule fragment description (see With Product ID in the screenshot above)
relatedStepType	<p>Template-based models are pre-defined with step types Recommend, Re-rank, and Filter. This parameter allows you to assign the rule fragment to a certain step type. The following step types are available:</p> <ul style="list-style-type: none"> ○ O = Offer recommendation ○ K = Re-rank ○ F = Filter ○ (Blank) = Recommend <p>⚠ Caution</p> <p>A rule fragment must be appropriate for a certain step type. For example, re-rank and filter rule fragments typically start with <code>Resultset.</code> as the data object that they process.</p>
objectClass	<p>Defines the object class for which this rule fragment is appropriate. Existing object classes are:</p> <ul style="list-style-type: none"> ○ PRODUCT ○ OFFER <p>⚠ Caution</p> <p>A rule fragment must be appropriate for a certain object class.</p>

Technical Component	Description
exampleResKey	Resource key suffix for the example description of the rule fragment (see Choose offers containing product ... in the screenshot above)
resultType	<p>Defines the result type for which this rule fragment is appropriate.</p> <p> Caution</p> <p>The result type must correspond to the one from the recommendation model type.</p>

Technical Component	Description
reskeyParamToToken	<p>Identifies which tokens (see description of <code>renderingData</code> in the SAP HANA Rules Development & Implementation Guide on SAP Service Marketplace at http://service.sap.com/instguides  <i>SAP In-Memory Computing</i>  <i>SAP HANA Rules Framework</i> ) are relevant for the rule fragment description (attribute <code>descriptionResKey</code>)</p> <p>It contains a comma-separated list of indices of <code>RenderingData</code> tokens that are parameters. For example, <code>"reskeyParamToToken": "1, 2"</code> defines that the tokens with the indices 1 and 2 in the token list of the <code>RenderingData</code> of an alias are parameters. For more information about <code>Alias</code>, <code>RenderingData</code>, and <code>Token</code>, see the HRF vocabulary definition). The idea is to extend the values of the <code>"reskeyParamToToken"</code> parameter to embed more information per parameter, in addition to its index. Especially for the length information the attribute <code>len:</code> is supported, such as <code>"reskeyParamToToken": "1, 2 (len:50)"</code>, which means:</p> <ul style="list-style-type: none"> Token 1 => no <code>len:</code> attribute given, no zero handling will be done Token 2 => field length = 50 characters, zero handling will be done <p>In the future, the list might be extended with more parameters, for example, <code>"reskeyParamToToken": "1 (len: 2; newParam:true), 2 (len:50; anotherParam:yyy)"</code>. The following key words are reserved for processing: <code>() len : ;</code></p> <p>For compatibility reasons, the comma-separated list of parameter tokens without length extension is still supported, for example, <code>"reskeyParamToToken": "1, 2"</code>. A combination of both is also allowed, for example, <code>"reskeyParamToToken": "1, 2 (len:50)"</code>.</p>
contextParams	Specifies which context parameters are available in the model preview. The context parameter is always of type <code>String</code> and consists of its ID and its <code>resourceKey</code> .
HRF alias <code>renderingData</code>	HRF native attribute that controls the rendering of rule fragments. For more information, see SAP Service Marketplace at http://service.sap.com/instguides  <i>SAP In-Memory Computing</i>  <i>SAP HANA Rules Framework</i> 

2. One or multiple rule fragments can form a business rule, in the sense of a recommendation. Based on an SAP HANA rules framework vocabulary (for example, [VocabularyPMR](#) in the diagram above), these rule fragments are stored as an SAP HANA rules framework rule on SAP HANA (for more information, see SAP HANA rules framework at <http://service.sap.com/instguides> [SAP In-Memory Computing](#) [SAP HANA Rules Framework](#)). The following example is from the SAP HANA rules framework:

Example

```
{ "description": "", "status": "Active", "ruleBody": { "type": "text", "content":  
  { "condition": "((Product.External_Id is equal to 'PRD-0002J 2' and  
    Product.Offer_Key = Offer.Offer_Key))" }, "vocabulary": "system-  
local.prodreco.hrf.vocabulary::recommendationPMR" }
```

The SAP HANA rules framework rules are not subject to extensibility, but only mentioned as part of the involved technical components.

3. When you preview or activate a recommendation model, the SAP HANA rules framework rules are compiled into a stored procedure, which is executable for a recommendation request (refers to the link [execute rules via](#) between the SAP HANA rules framework and views in the diagram above).

Rule Fragments and Views

To simplify rule fragment creation and optimize rule execution performance, rule fragments are built on top of SAP HANA view artefacts. So, rule fragment logic can be modelled into either the rule fragment itself or an SAP HANA view.

Eligibility

In the sense of offer handling, eligibility defines whether or not a business partner is allowed to order with reference to an offer, or even to see it.

Example

Table 27:

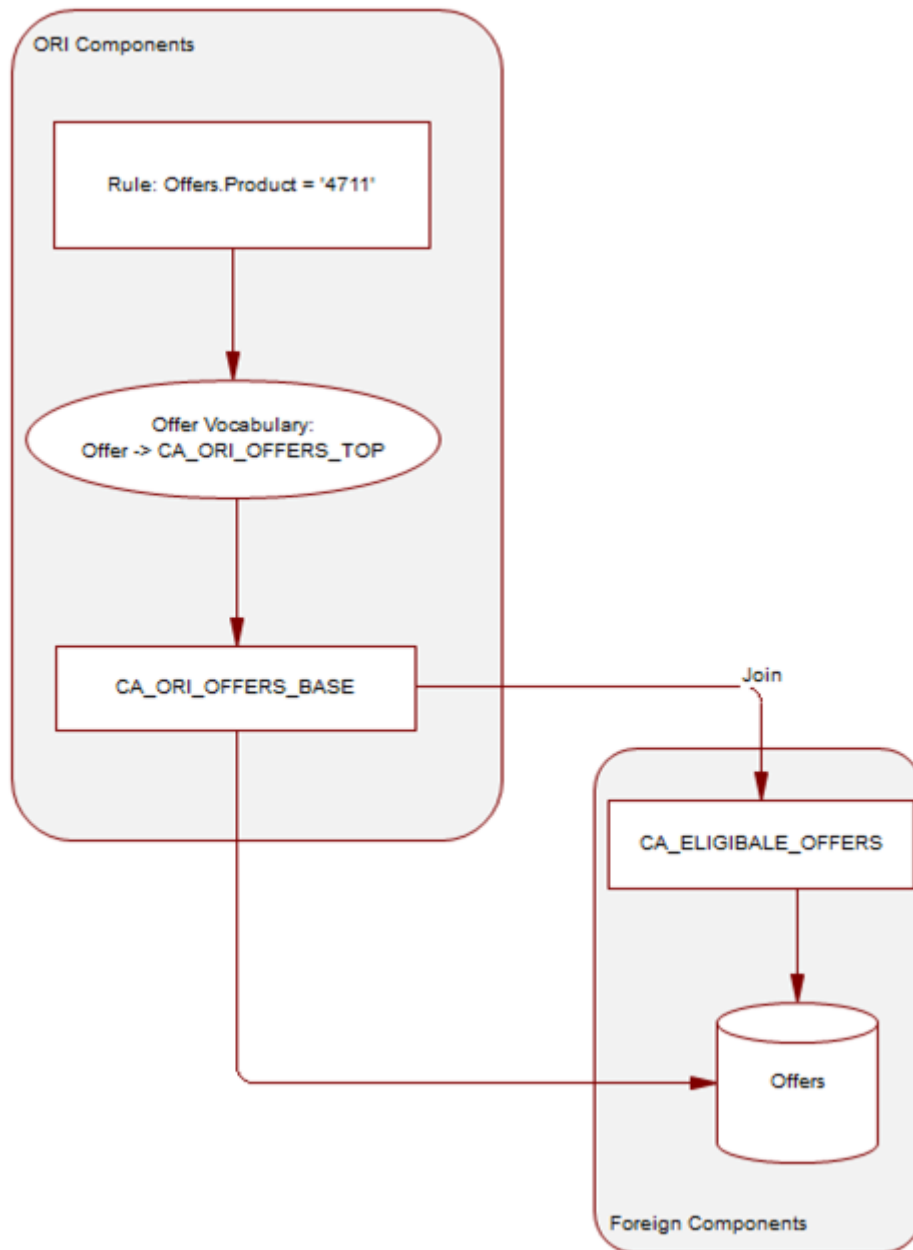
Offer	Trigger	Reward	Target Group
1	Product HT-1010	Discount 5 %	RareShopper
2	Product HT-1010	Discount 15 %	PremiumShopper

A business partner who is assigned only to target group `RareShopper` would not be allowed to order with reference to offer 2.

Eligibility Related to Recommendation Objects

Recommendation currently recognizes the two object types [Product](#) and [Offer](#). In general, a product recommendation does not have to deal with any eligibility checks, while for offers this is always necessary. So even from a technical perspective, it would not be an error to recommend an offer that a user would not be eligible for, but from a business perspective, this would be wrong. Recommending an offer that a user could not use in the order process (since the sales order would check for eligibility again and deny any ordering) would frustrate a user and therefore not make sense.

The following diagram shows a modelling proposal for eligibility-related SAP HANA views:



Level of Enhancements

Rule fragments that are delivered in the SAP standard delivery may not cover all business requirements. Therefore, recommendation allows enhancements to have custom rule fragments. Depending on the requirement, various objects need to be created again. The following sections guide you through the levels of enhancements. Additionally, we recommend that you become familiar with the SAP HANA rules framework. For

more information, see SAP Service Marketplace at <http://service.sap.com/instguides>  *SAP In-Memory Computing*  *SAP HANA Rules Framework*.

New Rule Fragment in Existing Attributes

All rule fragments are based on the HRF vocabulary. The HRF vocabulary, depending on your chosen data source during the technical configuration setup, includes various data objects and their attributes. The following SAP standard vocabularies are available and can be found under the main package `sap.hana-app.prodreco.rules.hrf.vocabulary` and main package `system-local.prodreco.extensibility.hrf.vocabulary`. *<data scenario related sub-package>*. *<vocabulary><>*

Table 28:

Description	Sub-Package	Name	Comment
Product-recommendation-related vocabulary intended for retail scenarios	prod	recommendationBase recommendationBaseAliases	
Offer-recommendation-related vocabulary intended for SAP Promotion Management for Retail (PMR) scenarios	pmr	recommendationPMR recommendationPMRAliases	
Offer-recommendation-related vocabulary intended for SAP Hybris Marketing offer scenarios	yMktgOffers	recommendationyMktOffr recommendationyMktOffrAliases	
Offer-recommendation-related vocabulary intended for combined SAP Hybris Marketing offer and SAP Promotion Management for Retail (PMR) scenarios	ymktpmr	recommendationyMktPMROffr recommendationyMktPMROffrAliases	This vocabulary is meant to serve as sample vocabulary in the event that you want to combine an offer object from outside of Hybris Marketing with the relevant offer object from Hybris Marketing that is meant to serve as the carrier of the digital marketing content. You need to verify the exact details of integration between these objects in accordance with the purpose, especially in terms of efficient access (performance) between the external offer object and the offer object from Hybris Marketing, and all relevant attributes of both objects.

These vocabularies in the `extension` area will be created only one by SAP installation procedure (technical configuration task lists `CUAN_SETUP_INITIAL` or `CUAN_SETUP_PRI`). The vocabularies are meant for customer enhancements and will not be overwritten anymore. Since the vocabularies are addressed by the algorithm default parameter customizing, entries will be immediately effective.

The SAP HANA rules framework standard tools can be used to simply browse the existing attributes. See section [From an HRF Rule to a Rule Fragment](#) below for information about how to design a rule fragment for recommendation. Once a new rule fragment is available, this must be registered in the [Snippet Repository](#) (see diagram above). The [Snippet Repository](#) is implemented as native HRF alias located in an HRF vocabulary. Its implementation is based on the new HRF SP7 functionality of `externalMetadata` and `renderingData`.

As a rule fragment not only consists of its technical content but also controlling information for the UI, see the detailed description above on how to build this information.

A new alias in this vocabulary will be immediately visible as a new rule fragment on the UI. It might be necessary to clear the browser cache if the rule fragment does not appear.

For simplification reason, the appropriate texts can be entered in metadata attributes `descriptionResKey` and `exampleResKey`. No enhancement of the web application is required here, but translation is then not possible.

Edit Rule Fragment

Example

Choose offers with ID >Premium<

OFFER_ID:

is equal to

PR101

OK

Delete

Cancel

If translation is required, the SAP Fiori application `hpa.cei.reco.mkt` (Hybris Marketing Recommendation Marketer UI) can be extended free of modifications, to add the new resource keys.

Table 29:

Resource Key Prefix
<code>UI.RULEEDITOR.SNIPPET.EXAMPLE.<exampleResKey></code>
<code>UI.RULEEDITOR.SNIPPET.DESCRPTION.<descriptionResKey></code>
<code><snippet id></code>




For more information about how to extend SAP Fiori applications, see https://help.hana.ondemand.com/SAP_RDE/frameset.htm?0a8397c350fd465583108f7575884db3.html.

Caution

Changes to rule fragments after they have been used in recommendation rules should be avoided, as this may lead to inconsistent behavior on the UI, and during generation.




















New Rule Fragment in New Attribute

Enhancing recommendations with a rule fragment that is based on new data object attributes requires the following steps:

1. Include the new data object attribute in a vocabulary. Create a new vocabulary that depends on another one (keep in mind that you want to transport your new vocabulary from a test system into a production system). Within this new vocabulary, use the SAP HANA rules framework extension mechanism to add a new data object attribute. For more information about how to create a new vocabulary, see SAP Service Marketplace at <http://service.sap.com/instguides>  [SAP In-Memory Computing](#)  [SAP HANA Rules Framework](#). 
2. Reference the vocabulary for recommendations. To do so, choose [Data Browser](#) (transaction **SE16**) and enter table name `PRECO_A_AL_DFT_P`. Enter `ALGORITHM_ID = RULES*`. Adapt the path to the new vocabulary.
3. Add the new rules fragment as described in the section [New Rule Fragment on Existing Attribute](#).

Integration of New Offer Database




Enhancing recommendations with a rule fragment that is based on a new data source requires the following steps.

1. Build your own vocabulary based on your new data source. For more information about how to create a new vocabulary, see SAP Service Marketplace at <http://service.sap.com/instguides>  [SAP In-Memory Computing](#)  [SAP HANA Rules Framework](#). 
2. Create your own data source type in Customizing at  [SAP Hybris Marketing](#)  [Recommendation](#)  [Data Source Definition](#)  [Define Data Source Types](#). 
3. Create your own data source in Customizing at  [SAP Hybris Marketing](#)  [Recommendation](#)  [Data Source Definition](#)  [Define Data Sources](#). 
4. Create your own algorithm in Customizing at  [SAP Hybris Marketing](#)  [Recommendation](#)  [Algorithm Definition](#)  [Define Algorithms](#).  Assign `RULE_S_VOCABULARY` as base algorithm parameter association.
5. Configure your new algorithm type and data source type as data type of result. Configure the type of implementation of the algorithm as rule.
6. Reference the vocabulary for recommendations. To do so, choose [Data Browser](#) (transaction **SE16**) and enter table name `PRECO_A_AL_DFT_P`. Create a new entry (for example, `GUID` could be created by executing function module `CRMOST_GUID_GET`): `Algorithm ID = <<your new algorithm>>; Parameter ID = RULE_S_VOCABULARY`.
7. Class `CL_PRECO_RUL_VOCABULARY_UTIL` controls the mapping of the leading object and the result type key. Method `BUILD_VOCABULARYDATA` allows you to add your own leading object and result type key, free of modifications. Use implicit enhancement options at the end of the method. For more information on implicit enhancement options, see http://help.sap.com/saphelp_nw75/helpdata/EN/29/e59441026aae5fe10000000a1550b0/content.htm .
8. Follow step 3 from the section [New Rule Fragment in New Attribute](#) above.

From an HRF Rule to a Rule Fragment

The easiest way to develop a new rule fragment is to start with a working SAP HANA rules framework rule. This rule may already have been tested with the available SAP HANA rules framework tools. Assume that an SAP HANA rules framework rule such as `EligibleOfferContent.OfferContentItemDetail.Offer.OfferName` is equal to *Premium*. This rule consists of the following components:

- Fix term `EligibleOfferContent.OfferContentItemDetail.Offer.OfferName`
- Operator *is equal to*
- Variable term *Premium*

These components must be entered as HRF alias `renderingData` according to the SAP HANA Rules Framework Development & Implementation Guide for alias `renderingData` on SAP Service Marketplace at <http://service.sap.com/instguides>  [SAP In-Memory Computing](#)  [SAP HANA Rules Framework](#) .





5.1.5 Consuming Recommendation Models Using Remote Function Call

Use

The following remote function calls (RFCs) enable customer channels to receive recommendations generated by Recommendation:

PROD_RECO_GET_RECOMMENDATIONS

You can use this RFC to provide consumers with recommendations for individual model types. The following parameters are required:

- `IV_APP_ANCHOR`
This parameter contains the ID of the application that consumes the recommendations provided by a model. For example, Segmentation. For more information about client applications, see Customizing for *SAP Hybris Marketing* under  [Data Management](#)  [Predictive Scenarios](#)  [Define Client Applications](#) .
- `IT_RECOMMENDERS`
This parameter is a table that stores a list of recommendation model types and associated parameters. The table contains the following:
 - `MODEL_TYPE`
The recommendation model type to use.
 - `LEADING_OBJECTS`
A table containing a list of leading objects. The table contains the following
 - `ITEM_TYPE`
The leading item type.
 - `ITEM_ID`
The leading item ID.
 - `BASKET_OBJECTS`
A table containing a list of objects in the consumer's cart. The table contains the following:
 - `ITEM_TYPE`
The leading item type.
 - `ITEM_ID`
The leading item ID.

- IT_CONTEXT_PARAMS

This parameter is a table that stores context information. The table contains the following:

- PREFILTER_PARAM_ID
The generated ID of the parameter.
- PARENT_PREFILTER_PARAM_ID
The parent parameter.
- PARAMETER_VALUE
The value of the parameter.
- PREFILTER_TYPE_ID
The prefilter parameter ID.

- IV_USER_ID

This parameter contains the user ID, for example, consumer ID.

- IV_USER_TYPE

This parameter contains the user type, for example, consumer.

The results are returned in the following output parameters:

- ET_RESULTS

This parameter is a table that stores the list of recommended products. The table contains the following:

- MODEL_TYPE
The model type that provided the recommendation.
- ITEM_ID
The item ID of a recommendation.
- ITEM_TYPE
The item type of a recommendation.
- SCORE
The score of a recommendation. The score reflects the value of the recommendation to the consumer; the higher the score the greater the value.

- ET_MESSAGES

This parameter is a table that stores the messages generated while retrieving the recommendations.

PROD_RECO_GET_RECO_BY_SCENARIO

You can use this RFC to provide consumers with recommendations for individual scenarios. The following parameters are required:

- IT_RECOMMENDERS

This parameter is a table that stores a list of recommendation model types and associated parameters. The table contains the following:

- SCENARIO_ID
The recommendation scenario to use.
- LEADING_OBJECTS
A table containing a list of leading objects. The table contains the following:
 - ITEM_TYPE
The leading generic object type.
 - ITEM_ID
The leading item ID.
- BASKET_OBJECTS
A table containing a list of objects in the consumer's cart. The table contains the following:
 - ITEM_TYPE

- `ITEM_ID`
The leading item ID.
- `IT_CONTEXT_PARAMS`
This parameter is a table that stores context information. The table contains the following:
 - `PREFILTER_PARAM_ID`
The generated ID of the parameter.
 - `PARENT_PREFILTER_PARAM_ID`
The parent parameter.
 - `PARAMETER_VALUE`
The value of the parameter.
 - `PREFILTER_TYPE_ID`
The prefilter parameter ID.
- `IV_USER_ID`
This parameter contains the user ID, for example, consumer ID.
- `IV_USER_TYPE`
This parameter contains the user as defined in the Origin of Interaction Contact ID.

The results are returned in the following output parameters:

- `ET_RESULTS`
This parameter is a table that stores the list of recommended products. The table contains the following:
 - `SCENARIO_TYPE`
The scenario providing the recommendation.
 - `ITEM_ID`
The item ID of a recommendation.
 - `ITEM_TYPE`
The generic object type of a recommendation.
 - `SCORE`
The score of a recommendation. The score reflects the value of the recommendation to the consumer; the higher the score the greater the value.
- `ET_MESSAGES`
This parameter is a table that stores the messages generated while retrieving the recommendations.

Example

The following is an example of an implementation that uses the `PROD_RECO_GET_RECOMMENDATIONS` RFC.

i Note

A context object must be prepopulated with the required information for the following implementation to be successful. For example, model type, item type and product ID, user ID, and cart items.

```
import sap.core.jco.connection.JCoConnection;
final String modelType = context.getRecommendationModelType();
final String itemType = context.getItemDataSourceType();
final String productId = context.getProductId();
try{
```

```

final JCoFunction function =
jCoConnection.getFunction("PROD_RECO_GET_RECOMMENDATIONS");
final JCoParameterList importParameterList = function.getImportParameterList();
final JCoTable recommenders = importParameterList.getTable("IT_RECOMMENDERS");
recommenders.appendRow();
recommenders.setValue("MODEL_TYPE", modelType);
final JCoTable leadingObjects = recommenders.getTable("LEADING_OBJECTS");
if ( productId != null &&
!productId.equals("") &&
!productId.equalsIgnoreCase("null")){
leadingObjects.appendRow();
leadingObjects.setValue("ITEM_TYPE", itemType);
leadingObjects.setValue("ITEM_ID", productId);
}
final JCoTable cartEntries = recommenders.getTable("BASKET_OBJECTS");
for (final String cartItem : context.getCartItems()){
cartEntries.appendRow();
cartEntries.setValue("ITEM_ID", cartItem);
cartEntries.setValue("ITEM_TYPE", itemType);
leadingObjects.appendRow();
leadingObjects.setValue("ITEM_TYPE", itemType);
leadingObjects.setValue("ITEM_ID", cartItem);
} importParameterList.setValue("IV_USER_ID", context.getUserId());
importParameterList.setValue("IV_USER_TYPE", configuration.getUserType());
jCoConnection.execute(function);
final JCoParameterList exportParameterList=function.getExportParameterList();
final JCoTable results = exportParameterList.getTable("ET_RESULTS");
if (!results.isEmpty()){
final int len = results.getNumRows();
for (int i = 0; i < len; i++){
results.setRow(i);
final String recommendationId = results.getString("ITEM_ID");
final String recommendationType = results.getString("MODEL_TYPE");
final ProductRecommendation productRecommendation =
createProductRecommendation(recommendationId, recommendationType);
if (productRecommendation != null) {
result.add(productRecommendation)
}
}
} catch (final sap.core.jco.exceptions.BackendException e){
LOG.error("", e);
}
return result;

```

5.1.6 Consuming Recommendation Models Using OData

Use

The PROD_RECO_RUNTIME_SRV OData service enables customer channels to receive recommendations generated by Recommendation. To receive the recommendations, call the following entities using the deep insert functionality of OData:

Root URL: `https://<Server>:<Port>/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/RecommendationScenarios`

Request Mode: POST

Permissions: PFCG role SAP_COM_CSR_0019

For more information about the deep insert functionality of OData, see <http://help.sap.com>. Choose ► [Technology](#) ► [SAP Gateway](#). Choose a release and then ► [Application Help](#). In SAP Library, choose ► [SAP NetWeaver Gateway Developer Guide](#) ► [OData Channel](#) ► [Advanced Features](#) ► [Deep Insert](#).

The nested structure of the entities that can be navigated to from the `RecommendationScenarios` entity are as follows:

- `RecommendationScenarios`
 - `Scenarios`
 - `LeadingObjects`
 - `BasketObjects`
 - `ContextParams`
 - `ScenarioHashes`
 - `ResultObjects`

RecommendationScenario Entity Parameters

The following table contains the parameters of the `RecommendationScenario` entity:

Table 30:

Property	Description	Edm Core Type	Max Length	Key
<code>UserId</code>	The ID of the user who performs the interaction, for example, customer ID or contact ID.	Edm.String	50	TRUE
<code>UserType</code>	The type of user who performs the interaction. For more information, see Customizing for ► SAP Hybris Marketing ► Recommendation ► Data Source Definition ► Define Data Source Types (Origin of Contact) .	Edm.String	20	TRUE
<code>ExternalTracking</code>	A flag that implies external tracking of impressions using the <code>PostImpressions</code> function import (Optional).	Edm.Boolean	1	FALSE

i Note

When the `ExternalTracking` parameter in the `RecommendationScenario` entity is set to `TRUE`, SAP Hybris Marketing Cloud will not count the impressions for the recommendation scenario that is being solicited. To keep the number of impressions in SAP Hybris Marketing Cloud accurate, it is necessary for the external system to convey the impression count using the `PostImpressions` function import. For more information, see [Providing Impressions Data Using Function Import \[page 138\]](#).

Scenario Entity Parameters

The following table contains the parameters of the `Scenario` entity:

Table 31:

Property	Description	Edm Core Type	Max Length	Key
<code>ScenarioId</code>	The scenario ID represents a model type and related usage information, for example, promotion model type and user type. You can define the scenario ID in Recommendation Scenarios .	Edm.String	50	TRUE
<code>HashId</code>	A hash associated to a specific user. The hash accelerates retrieving recommendations from the cache of an optimized algorithm.	Edm.String	32	FALSE

LeadingObject Entity Parameters

The following table contains the parameters of the `LeadingObject` entity:

Table 32:

Property	Description	Edm Core Type	Max Length	Key
<code>LeadingObjectId</code>	The ID of the leading object, for example, material number.	Edm.String	50	TRUE

Property	Description	Edm Core Type	Max Length	Key
LeadingObjectType	A recommendation data source type that is defined to an ITEM-data source class. For more information, see Customizing for ▶ SAP Hybris Marketing > Recommendation > Data Source Definition > Define Data Source Types (Generic Object Type) ▶	Edm.String	30	TRUE

BasketObjectId Entity Parameters

The following table contains the parameters of the `BasketObjectId` entity:

Table 33:

Property	Description	Edm Core Type	Max Length	Key
BasketObjectId	The ID of the leading object, for example, material number.	Edm.String	50	TRUE
BasketObjectType	A recommendation data source type that is defined to an ITEM-data source class. For more information, see Customizing for ▶ SAP Hybris Marketing > Recommendation > Data Source Definition > Define Data Source Types (Generic Object Type) ▶	Edm.String	30	TRUE

ContextParam Entity Parameters

The following table contains the parameters of the `ContextParam` entity:

Table 34:

Property	Description	Edm Core Type	Max Length	Key
<code>ContextId</code>	The prefilter parameter ID.	Edm.Int32	n.a.	TRUE
<code>ContextParamId</code>	The parent prefilter parameter ID.	Edm.Int32	n.a.	FALSE
<code>Value</code>	The value of the prefilter parameter.	Edm.String	100	FALSE
<code>ValueType</code>	The value type of the prefilter parameter.	Edm.String	32	FALSE

ScenarioHashes Entity Parameters

The following table contains the parameters of the `ScenarioHashes` entity:

Table 35:

Property	Description	Edm Core Type	Max Length	Key
<code>ScenarioId</code>	The recommendation scenario ID.	Edm.String	50	TRUE
<code>HashID</code>	A hash returned by the system that is associated to a specific user. The hash accelerates retrieving recommendations from the cache of an optimized algorithm.	Edm.String	32	TRUE
<code>ExpiresOn</code>	Expiry date of HashID.	Edm.DateTime		FALSE
<code>ResultScope</code>	The scope of the result. For example, Generic, Restricted, or Personalized.	Edm.String	1	FALSE

ResultObject Entity Parameters

The following table contains the parameters of the `ResultObject` entity:

Table 36:

Property	Description	Edm Core Type	Max Length	Key
<code>ScenarioId</code>	The recommendation scenario ID.	Edm.String	50	TRUE
<code>ResultObjectType</code>	A recommendation data source type that is defined to an ITEM-data source class. For more information, see Customizing for SAP Hybris Marketing Recommendation Data Source Definition > Define Data Source Types (Generic Object Type) >	Edm.String	30	TRUE
<code>ResultObjectId</code>	The ID of the result object, for example, material number.	Edm.String	50	TRUE
<code>ResultObjectScore</code>	The score of the result object.	Edm.Decimal	10.5	FALSE

Example

HTTP Post Request Using Deep Insert Functionality of OData in JSON Encoding:

```
{
  "UserId" : "40F2E9306E391ED59BDE581AFE71F329 ",
  "UserType" : "COOKIE_ID",
  "ExternalTracking" : false,
  "Scenarios" :
  [
    {
      "ScenarioId" : "INT_TEST",
      "HashId" : "D33DD1F71615D50334FB2F1043365430",
      "LeadingObjects" :
      [
        {
          "LeadingObjectType" : "SAP_ERP_MATNR",
          "LeadingObjectId" : "M-01"
        }
      ],
      "BasketObjects" :
      [
        {
          "BasketObjectType" : "SAP_ERP_MATNR",
          "BasketObjectId" : "100-100"
        }
      ]
    }
  ],
}
```

```

"ContextParams" : [],
"ScenarioHashes" : [],
"ResultObjects" : []
}

```

HTTP Post Response Payload in JSON Encoding :

```

{
  "d": {
    {
      "__metadata": {
        "id": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/RecommendationScenarios(UserId='',UserType='')",
        "uri": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/RecommendationScenarios(UserId='',UserType='')",
        "type": "PROD_RECO_RUNTIME_SRV.RecommendationScenario"
      }
      "UserId": ""
      "UserType": ""
      "ExternalTracking": true,
      "ScenarioHashes": {
        "results": [
          {
            "__metadata": {
              "id": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ScenarioHashes('SAP_TOP_SELLERS_EMAIL_CAMPAIGN')",
              "uri": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ScenarioHashes('SAP_TOP_SELLERS_EMAIL_CAMPAIGN')",
              "type": "PROD_RECO_RUNTIME_SRV.ScenarioHash"
            },
            "ScenarioId": "SAP TOP SELLERS EMAIL CAMPAIGN"
            "HashId": "D33DD1F71615D50334FB2F1043365429",
            "ExpiresOn": "/Date(1478180969524)/",
            "ResultScope": "G"
          },
        ]
      },
      "ResultObjects": {
        "results": [3]
        0: {
          "__metadata": {
            "id": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='100-100')",
            "uri": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='100-100')",
            "type": "PROD_RECO_RUNTIME_SRV.ResultObject"
          }
          "ScenarioId": "INT_TEST"
          "ResultObjectType": "SAP_ERP_MATNR"
          "ResultObjectId": "100-100"
          "ResultObjectScore": "1.00000"
        }
        1: {
          "__metadata": {
            "id": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/

```

```

ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='
P-102')"
    "uri": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/
PROD_RECO_RUNTIME_SRV/
ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='
P-102')"
    "type": "PROD_RECO_RUNTIME_SRV.ResultObject"
  }
  "ScenarioId": "INT_TEST"
  "ResultObjectType": "SAP_ERP_MATNR"
  "ResultObjectId": "P-102"
  "ResultObjectScore": "0.01906"
}
2:
{
  "__metadata":
  {
    "id": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/
PROD_RECO_RUNTIME_SRV/
ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='
P-100')"
    "uri": "https://[sap-hybris -marketing-server]/sap/opu/odata/sap/
PROD_RECO_RUNTIME_SRV/
ResultObject(ScenarioId='INT_TEST',ResultObjectType='SAP_ERP_MATNR',ResultObjectId='
P-100')"
    "type": "PROD_RECO_RUNTIME_SRV.ResultObject"
  }
  "ScenarioId": "INT_TEST"
  "ResultObjectType": "SAP_ERP_MATNR"
  "ResultObjectId": "P-100"
  "ResultObjectScore": "0.00554"
}
}
}
}
}

```

5.1.6.1 Enabling Value Help Entities

Entities that enable you to choose recommendation scenario and item source type parameters from a value help.

The PROD_RECO_RUNTIME_SRV OData service enables customer channels to receive recommendations generated by Recommendation. The RecommendationRecoScenarios and ItemSourceTypes entities enable customer channels to choose ScenarioID, LeadingObjectType, or BasketObjectType parameters from a value help.

ProductRecoScenarios Entity

Root URL: `https://<Server>:<Port>/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ProductRecoScenarios`

Request Mode: GET

Permission: PFCG role SAP_COM_CSR_0019

ProductRecoScenario Entity Parameters

The following table contains the parameters of the `ProductRecoScenario` entity:

Table 37:

Property	Description	Edm Core Type	Max Length	Key
ScenarioId	The ID of the scenario.	Edm.String	50	TRUE
ScenarioDescription	The description of the scenario.	Edm.String	255	FALSE
Language	The language of the scenario description.	Edm.String	30	FALSE

ItemSourceTypes Entity

Root URL: `https://<Server>:<Port>/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/ItemSourceTypes`

Request Mode: GET

Permission: PFCG role `SAP_COM_CSR_0019`

ItemSourceTypes Entity Parameters

The following table contains the parameters of the `ItemSourceTypes` entity:

Table 38:

Property	Description	Edm Core Type	Max Length	Key
ItemSourceId	The ID of the item source.	Edm.String	2	TRUE
ItemSourceTypeDescription	The description of the item source type.	Edm.String	255	FALSE
ItemSourceObjectType	The object type of the item source.	Edm.String	30	FALSE

5.1.7 Providing Impressions Data Using Function Import

The `PostImpressions` function import enables external tracking of Impressions.

The `PROD_RECO_RUNTIME_SRV` OData service enables customer channels to receive recommendations generated by Recommendation. When the `ExternalTracking` parameter in the `RecommendationScenario` entity is set to `TRUE`, SAP Hybris Marketing Cloud will not count the impressions for the recommendation scenario that is being solicited. To keep the number of impressions in SAP Hybris Marketing Cloud accurate, it is necessary for the external system to convey the impression count using the `PostImpressions` function import.

The following table contains the parameters of the `PostImpressions` function import:

Table 39:

Property	Description	Edm Core Type	Max Length
ScenarioId	The recommendation scenario ID.	Edm.String	50
TimeStamp	The timestamp of the impression.	Edm.DateTimeOffset	30
ImpressionCount	The total number of impression performed.	Edm.Int16	
ItemCount	The total number of Item recommended.	Edm.Int16	



Example

```
https://[sap-hybris -marketing-server]/sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/  
PostImpressions?ScenarioId='  
INT_TEST'&TimeStamp=datetimeoffset'2016-12-03T12:45:29Z'&ImpressionCount=1&ItemCount  
=3&saml2=disabled
```

5.1.8 Providing Interactions Data Using Remote Function Call

Use

Recommendation enables host systems to post interactions to SAP HAHA using remote function calls (RFCs), and later, consume the information in a recommendation model as a data source. An interaction can be any event performed by a consumer on a Web shop. For example, you can post any or all of the following interactions:

- Click through
Consumer clicks on a recommended item to view its details.
- Acceptance
Consumer adds a recommended item to basket.
- Conversion
Consumer purchases a recommended item.
- Business event view
Consumer clicks on a product (not recommended) to view its details.

The following RFCs are available:

PROD_RECO_POST_INTERACTION

The following parameters are required:

- IT_INTERACTIONS

This parameter is a table that stores a list of interactions to be posted to SAP Hybris Marketing. The table contains the following:

- `USER_TYPE`
The product recommendation data source type with a `USER` data source class. For more information about data source types, see Customizing for [SAP Hybris Marketing](#) under [► Recommendation ► Data Source Definition ► Define Data Source Types ►](#)
- `USER_ID`
The ID of the user who performs the interaction, for example, customer ID or contact ID.
- `COMM_MEDIUM`
The communication medium. For more information about the communication medium, see Customizing for [SAP Hybris Marketing](#) under [► Data Management ► Interactions ► Define Communication Media ►](#)
- `IA_TYPE`
The Interaction type. For more information about the interaction type, see Customizing for [SAP Hybris Marketing](#) under [► Data Management ► Interactions ► Define Interaction Types ►](#)
- `SOURCE_SYSTEM_TYPE`
The source system, for example, a Web shop or a register.
- `SOURCE_SYSTEM_ID`
The ID of the source system.
- `SOURCE_OBJECT_ID`
The ID of the session the user performed the interaction in.
- `TIMESTAMP`
The coordinated universal time (UTC) stamp of when the interaction happened.
- `PRODUCTS`
This parameter is a table that contains the items involved in the interaction. The table contains the following:
 - `OBJECT_TYPE`
The product Recommendation data source type with an `ITEM` data source class. It is defined in Customizing for [SAP Hybris Marketing](#) under [► Recommendation ► Data Source Definition ► Define Data Source Types ►](#)
 - `OBJECT_ID`
The ID of the item, for example, material number.
 - `PRECO_MODEL`
The model ID of the recommender system that recommends the item, for example, CROSSSELL.
 - `PRECO_SYSTEM`
The recommender system, for example, SAP Hybris Marketing Recommendation.
- `ET_FAILED_INTERACTIONS`
This parameter is a table that stores a list of interactions that could not be posted to SAP Hybris Marketing. The table contents are identical to the `IT_INTERACTIONS` parameter.
- `ET_MESSAGES`
This parameter is a table that stores the messages generated during the posting of the interactions to SAP Hybris Marketing.

PROD_RECO_POST_IA_FOR_SCENARIO

The following parameters are required:

- `IT_INTERACTIONS`

This parameter is a table that stores a list of interactions to be posted to SAP Hybris Marketing. The table contains the following:

- `SCENARIO_ID`
The ID of the user who performs the interaction, for example, customer ID or contact ID.
- `USER_ID`
The ID of the user who performs the interaction, for example, customer ID or contact ID.
- `IA_TYPE`
The Interaction type. For more information about the interaction type, see Customizing for [SAP Hybris Marketing](#) under [Data Management](#) > [Interactions](#) > [Define Interaction Types](#) .
- `SOURCE_OBJECT_ID`
The ID of the session the user performed the interaction in.
- `TIMESTAMP`
The coordinated universal time (UTC) stamp of when the interaction happened.
- `IPRODUCTS`
This parameter is a table that contains the items involved in the interaction. The table contains the following:
 - `OBJECT_TYPE`
The generic object type of a recommendation.
 - `OBJECT_ID`
The ID of the item, for example, material number.
 - `PRODUCT_QUANTIFIER`
A quantifier
 - `PRODUCT_IMAGE_URL`
A URL to an image.
 - `PRODUCT_NAV_URL`
A URL to a web page that contains the product. For example, your web shop.
 - `PRODUCT_DESC`
A description of the product.
- `ET_FAILED_INTERACTIONS`
This parameter is a table that stores a list of interactions that could not be posted to SAP Hybris Marketing. The table contents are identical to the `IT_INTERACTIONS` parameter.
- `ET_MESSAGES`
This parameter is a table that stores the messages generated during the posting of the interactions to SAP Hybris Marketing.

Example

The following is an example of an implementation that uses the `PROD_RECO_POST_INTERACTION` RFC.

i Note

The following example is written in JAVA and uses an SAP Java Connector (JCo) library. A recommendation context object must be prepopulated with the required information for the following implementation to be successful. For example, user type, user ID, session ID, product type, product ID, and recommendation model.

```
import sap.core.jco.connection.JCoConnection; import
sap.core.jco.exceptions.BackendException; public void
handleProductRecommendationEvent(final RecommendationContext context) { final
JCoConnection jCoConnection = getDefaultJCoConnection(); try { final JCoFunction
function = jCoConnection.getFunction("PROD_RECO_POST_INTERACTION"); final
JCoParameterList importParameterList = function.getImportParameterList(); final
JCoTable interactions = importParameterList.getTable("IT_INTERACTIONS");
interactions.appendRow(); interactions.setValue("USER_TYPE",
context.getUserType()); interactions.setValue("USER_ID", context.getUserId());
interactions.setValue("COMM_MEDIUM", "WEB"); interactions.setValue("IA_TYPE",
"CLICK_THROUGH"); interactions.setValue("SOURCE_SYSTEM_TYPE", "WEB_SHOP");
interactions.setValue("SOURCE_SYSTEM_ID", context.getStoreId());
interactions.setValue("SOURCE_OBJECT_TYPE", "WEB_SESSION");
interactions.setValue("SOURCE_OBJECT_ID", context.getSessionId());
interactions.setValue("TIMESTAMP", new Timestamp(new java.util.Date().getTime()));
final JCoTable products = interactions.getTable("PRODUCTS"); products.appendRow();
products.setValue("OBJECT_TYPE", context.getProductType());
products.setValue("OBJECT_ID", context.getProductId());
products.setValue("PRECO_MODEL", context.getRecommendationModel());
products.setValue("PRECO_SYSTEM", "PRI"); jCoConnection.execute(function); } catch
(final BackendException e) { LOG.error("", e); } }
```

5.1.9 Providing Interaction Data Using OData

Use

The `PROD_RECO_RUNTIME_SRV` OData service enables host systems to post interactions to an SAP HANA database, and then consume the information in a recommendation model. An interaction can be any event performed by a consumer on a Web shop. To post interactions, call the following entities using the deep insert functionality of OData:

- Interactions

The following are properties of a Interactions entity:

- ScenarioId
The recommendation scenario ID represents a model type and related usage information, for example, promotion model type and user type. It is defined in *SAP Hybris Marketing* in *Recommendation Scenario*.
- UserId
The ID of the user who performs the interaction, for example, customer ID or contact ID.
- InteractionType
The interaction type, for example, click through and acceptance. For more information, see Customizing for *SAP Hybris Marketing* under **Data Management > Interactions > Define Interaction Types**.
- Timestamp
The coordinated universal time (UTC) stamp of when the interaction happened.
- SourceObjectId
The ID of the session the user performed the interaction in.

- InteractionItems

The following are properties of a InteractionItems entity:

- ItemType
A recommendation data source type that is defined to a [Item](#) data source class. For more information, see Customizing for *SAP Hybris Marketing* under ► [Recommendation](#) ► [Data Source Definition](#) ► [Define Data Source Types](#) .
- ItemId
The ID of the item, for example, material number.

For more information about the deep insert functionality of OData, see <http://help.sap.com> . Choose ► [Technology](#) ► [SAP Gateway](#) . Choose a release and then ► [Application Help](#) . In SAP Library, choose ► [SAP NetWeaver Gateway Developer Guide](#) ► [OData Channel](#) ► [Advanced Features](#) ► [Deep Insert](#) .

Example

HTTP Post Request Using Deep Insert Functionality of OData

Request URL: http://sap/opu/odata/sap/PROD_RECO_RUNTIME_SRV/Interactions

Payload in JSON format:

```
{
  "ScenarioId" : "B2BWEB",
  "InteractionType": "CLICK_THROUGH",
  "TimeStamp": "2014-10-24T00:00:00Z",
  "SourceObjectId": "123456789",
  "InteractionItems" : [
    { "ItemType" : "1",
      "ItemId" : "M-05"},
    { "ItemType" : "1",
      "ItemId" : "M-01" }
  ]
}
```

The HTTP post response does not contain any entity.

5.2 Offers

5.2.1 Custom Fields for Offer Header and Offer Content

Offer extensibility scenario allows you to extend offer header (root) and offer content entities and includes automatic generation of respective metadata annotations and the correct data provision based on these metadata.

With SAP Hybris Marketing you have the following possibilities to extend your solution:

- Use ABAP Dictionary (transaction SE11) for the Business Object Processing Framework (BOPF), OData, and the SAP HANA Studio for adapting the HANA views.
- Choose directly the *Custom Fields and Logic* app to extend your fields and then bring the fields on the user interface (UI) by choosing the Customizing for SAP Hybris Marketing under ► *Offers* ► *Define Offer Content Types* ►

Using the Backend, SAP HANA Studio and Customizing

- **BOPF (DDIC) and OData**

To extend the structures for offer header (root) and offer content that are used in BOPF and OData, you can use, for example, the transaction SE11:

- CUAN_S_OFM_OFFER_UI
Extension includes (BOPF and OData) for offer header (root) are:
INCL_EEW_CUAN_OFM_ROOT_D for persistent data
INCL_EEW_CUAN_OFM_ROOT_TD for transient data
- CUAN_S_OFM_CONTENT_UI
Extension includes (BOPF and OData) for offer content are:
INCL_EEW_CUAN_CONTENT_ITMDET_D for persistent data
INCL_EEW_CUAN_CONTENT_ITMDT_TD for transient data

i Note

Please create an own APPEND for each extension structure before adding the new fields. Extension fields then will be assigned to these appends.

The customer namespace for the fields should have a prefix such as **z_** or as in the upcoming references **yy1_**.

Supported field types for the user interface (UI) are:

Table 40:

UI Field Type	Domain as Technical Reference
Amount	CUAN_MSM_AMOUNT
Currency	WAERS
Quantity	MENGVS

UI Field Type	Domain as Technical Reference
Unit of Measure	MEINS
Checkbox	BOOLE
List	CHAR<length> (1)
Number	DEC<length>,<decimals> (2)
Text	CHAR<length> (1)
Date	DATS
Time	TIME

(1) different length

(2) different length and count decimals

To add customer fields, such as YY1_CONTADD1_MOF in extension includes you have to use the following pattern:

- Offer Header (Root): `<namespace>_<fieldname>_MOF` (`_MOF` as suffix)
- Offer Content: `<namespace>_<fieldname>_MOC` (`_MOC` as suffix)

Keep in mind that different (extension) field types need different additional fields containing specific information, these additional fields are created with specific suffixes:

Table 41:

Suffix	For Include Type	Comment
F	Transient	Field control field that is added for any field type and defines the visibility of the field on instance level.
T	Transient	Text field that contains the description for code fields and is used for field type List, Amount and Quantity.
C	Persistent	Currency field is used for amounts.
U	Persistent	Unit of measure field used for quantities

For a complete extension the following fields are mandatory:

- Each extension field shall end with suffix `_MOFF` or `_MOCF`, has to be added to transient extension include, and shall have a field control field for metadata with *Component Type* (data element) `HPA_UX_FC_ALL`.
- Add a text field with suffix `_MOFT` or `_MOCT` to transient extension include, if a text shall be shown as field label. This could come from a value table, search help or domain fixed value or as a text for currency or quantity. Assign the *Component Type* `CHAR` to the field and enter a length value as required: In case of reference to a currency field `CHAR` length 15, in case of reference to amount field `CHAR` length 10 is sufficient.

- Add a currency field with suffix **_MOFC** or **_MOCC** to persistent extension include as currency reference for amount fields. Normally currency fields are combined with field control fields and text fields and the technical reference is *Domain* WAERS.
- Add a unit of measure field with suffix **_MOFU** or **_MOCU** to persistent extension include as unit reference for quantity fields. Normally currency fields are combined with field control fields and text fields and the technical reference is *Domain* MEINS.

Note

The part in front of the suffix must be identical for a set of fields!

The following suffix combinations for a complete set of fields are valid:

Table 42:

Suffix	Field Type of _MOF/ _MOC Field
F	Mandatory for each field type
F, T	List field
F, T, C	Amount field
F, T, U	Quantity field

Save and activate the corresponding DDIC-objects after adding the required combination of fields for extension.

• SAP HANA View

Now you have to adapt the SAP HANA view `AT_SEARCH_OFFER` for offer header (root) extension, respectively `AT_SEARCH_OFFER_CONTENT` for offer content extension with mass copy functionality in SAP HANA Studio.

Path: `sap.hana-app.cuan.offer.internal`

For more information, see [Extending the Data Source and Business Objects \[page 16\]](#).

• Checks and Usage

Now you can check your extensions with regards to OData and annotations: Retrieve the metadata, for example, with SAP Gateway Client (transaction `/IWFND/GW_CLIENT`) and check whether generic properties and annotations are visible: The YY1-fields of the extension structures are displayed as properties of the regarding entity (offer or offer content).

In addition they are linked to corresponding field control fields (`sap:field-control=`), text (`sap:text=`) respectively unit or currency fields (`sap:unit=`).

In case of quantity or amount fields also the corresponding unit of measure (`sap:semantics= "unit-of-measure"`) and currency (`sap:semantics= "currency-code"`) field metadata are specified:

Sample Code

```
<Property Name="YY1_CONTDAT_MOC" Type="Edm.DateTime" sap:label="Date"
Precision="0" sap:is-extension-field="true" sap:field-
control="YY1_CONTDAT_MOCF" sap:display-format="Date"/>
<Property Name="YY1_CONTCH10_MOC" Type="Edm.String" sap:label="Character Field
Length = 10" MaxLength="10" sap:is-extension-field="true" sap:field-
control="YY1_CONTCH10_MOCF"/>
<Property Name="YY1_AMOUNT_MOC" Type="Edm.Decimal" sap:label="Spend Amount"
Precision="16" sap:is-extension-field="true" sap:text="YY1_AMOUNT_MOCT"
sap:field-control="YY1_AMOUNT_MOCF" sap:unit="YY1_AMOUNT_MOCC" Scale="3"/>
```

```
<Property Name="YY1_AMOUNT_MOCC" Type="Edm.String" sap:label="Spend Currency"
MaxLength="5" sap:is-extension-field="true" sap:field-
control="YY1_AMOUNT_MOCF" sap:semantics="currency-code"/>
```

Then you can find the fields also in the Customizing for SAP Hybris Marketing under *Offers*:

- Header (root) extension fields in IMG activity ► [Arrange Extension Fields for the Header Data on the UI](#) ► [Assign Field to Field Group](#) ►.
- Offer content extension fields in IMG activity ► [Define Offer Content Types](#) ► [Assign Field to Field Group](#) ►.

As a final step the fields can be configured using Customizing for the respective UI's and afterwards tested in the *Offer* app whether they behave as expected.

- **Remove Extensions**

To remove extension fields, you must do the following steps in the specified sequence, otherwise inconsistencies can result in problems during offer processing:

1. Remove field from respective customizing.
2. Remove field from respective SAP HANA view: AT_SEARCH_OFFER for offer root extension or AT_SEARCH_OFFER_CONTENT for offer content extension.
Path: sap.hana-app.cuan.offer.internal
3. Remove field from appends.

Check: *Offer* app runs smoothly, respective fields do not appear anymore.

Using the Apps and Customizing

- **Extend the Offer**

1. Choose the *Custom Fields and Logic* app.
2. To get at first overview about existing fields for the offer set the filter for *Business Context* to **Marketing: Offer Header Data** and **Marketing: Offer Content Data**.
3. To create a new field choose *Create*.
4. In the popup *New Field* choose a suitable *Business Context* for offer, such as **Marketing: Offer Header Data** or **Marketing: Offer Content Data**.
5. Enter a *Label* for the new field on the respective screen. The entered data will be propagated to the fields *Identifier* and *Tooltip* as a proposal but can be changed. The *Identifier* is the name of the component in the database. This value is enhanced by a specific prefix and suffix (values shown in front/behind the input field).
6. Finally chose a *Type* of field from the value help.
In some cases additional parameters have to be maintained, for example, the field length or values in case of type **List**.
7. For saving choose *Create and Edit*.
8. Under *UIs and Reports* you have to enable the field usage for the respective UI or report.
9. When all required settings are made choose *Publish* to make the new field available.

- **Change Extension**

You can change labels and tooltips, and enable/disable the usage. After a change you must publish the fields again.

- **Delete an Extension**

You can simply delete extension fields again, but keep in mind that you remove all references to these fields beforehand.

- **Bring the Fields on the UI**

To bring the fields now on the UI choose the Customizing for SAP Hybris Marketing under *Offers* and work through the activities *Arrange Extension Fields for the Header Data on the UI* and *Define Offer Content Types*.

5.2.2 Reporting For Displayed and Clicked Offers Out of External Systems

If you display your created offers from SAP Hybris Marketing also in an external system, such as a web shop, you have the possibility to do a reporting about the displayed and clicked offers within SAP Hybris Marketing.

To do so you must connect the external system using an OData service `CUAN_IMPORT_SRV` with the SAP HANA report `sap.hana-app.cuan.offer.external:CA_OFFER_IA_REPORT` using SQL. SAP offers the interaction types `OFFER_DISPLAY` and `OFFER_CLICK` for reporting.

If a customer then gets an offer displayed or clicks on an offer the system tracks the interactions.

For more information about the OData service `CUAN_IMPORT_SRV`, see the current Data Management Upload Interfaces SAP Hybris Marketing on the SAP Service Marketplace <http://service.sap.com/mkt>.

5.3 Accessing Multichannel Transaction Data

Note

This section explains possible interactions between SAP Hybris Marketing, SAP Customer Activity Repository, and SAP CRM. Please note that SAP Customer Activity Repository and SAP CRM must be licensed, installed, and configured separately, and that your SAP license for SAP Hybris Marketing does not cover their acquisition or use.

If your system landscape includes SAP Customer Activity Repository, and if both SAP Hybris Marketing and SAP Customer Activity Repository are connected to the same SAP HANA database, you can access transaction data from multiple sales channels. SAP Customer Activity Repository collects transaction information from the interaction center in SAP Customer Relationship Management, and from retail stores. With this additional information available to SAP Hybris Marketing, you can further refine the target groups that you create with Segmentation.

Prerequisites

You have performed the following tasks:

Table 43:

Task	More Information
Installed SAP Customer Activity Repository and performed the steps required to integrate it with SAP CRM	<ul style="list-style-type: none">Installation Guide for SAP Customer Activity Repository, available on SAP Service Marketplace at http://service.sap.com/instguides, under ► <i>Installation and Upgrade Guides</i> ► <i>Industry Solutions</i> ► <i>Industry Solution Guides</i> ► <i>SAP for Retail</i> ► <i>SAP Customer Activity Repository</i>.SAP Library for SAP Customer Activity Repository on SAP Help Portal at http://help.sap.com/car. Choose a release and then <i>Application Help</i>. In SAP Library, choose <i>SAP HANA Content for SAP Customer Activity Repository</i> ► <i>Other Content</i> ► <i>Multichannel Sales Repository</i> ► <i>Integration with SAP Customer Relationship Management</i>.
Configured SAP Customer Activity Repository to handle multi-channel sales	SAP Library for SAP Customer Activity Repository on SAP Help Portal at http://help.sap.com/car . Choose a release and then <i>Application Help</i> . In SAP Library, choose <i>Multichannel Transaction Data Management</i> .
Set up customer identification in SAP Customer Activity Repository	SAP Library for SAP Customer Activity Repository on SAP Help Portal at http://help.sap.com/car . Choose a release and then <i>Application Help</i> . In SAP Library, choose <i>POS Transaction Management</i> ► <i>Task Processing</i> ► <i>Tasks for Enhancing Transactions</i> ► <i>Enhancing POS Transaction Data with Customer Numbers</i> .

Procedure

To configure SAP Hybris Marketing to work with SAP Customer Activity Repository, you must perform the following steps in SAP Hybris Marketing:

1. Specify new data source aliases in Customizing under ► *SAP hybris Marketing* ► *Segmentation* ► *Define Aliases for SAP HANA Data Sources*. Add the following entries to the table:

Table 44:

Data Source Alias	Data Source Location
SAP_MCF_BP	sap.is.retail.car/CRM_BUSINESS_PARTNER_INFO

Data Source Alias	Data Source Location
SAP_MCF_MCSR	sap.is.retail.car/MULTICHANNEL_BUSINESS_PARTNER_SALE_ITEM

- In *Business Configuration Sets: Activation* (transaction SCPR20), activate BC Set for ADT Settings in Customer Activity Repository (CEI_ADT_CAR).
This is a business configuration (BC) set that resides in SAP Hybris Marketing, and that automatically implements Customizing settings and values required to access data from SAP Customer Activity Repository.

6 SAP Hybris Marketing Planning

6.1 Programs

In the [Programs](#) application, marketing managers and marketing experts can create programs, which are containers for campaigns, and propose how much to spend on these programs. The [Programs](#) application can be extended.

The [Programs](#) application can be extended by adding custom fields to the program header.

6.1.1 Custom Fields in Programs

Custom fields can be added to the program header in the [Programs](#) application.

You can use the [Custom Fields and Logic](#) application to extend the fields for the program header and then bring the fields to the [Programs](#) application by adapting the user interface (UI) at runtime.

Creating Custom Fields

1. In the [Custom Fields and Logic](#) application, to get an overview of the existing fields for programs, set the filter for the business context to **Marketing Program**.
2. To create a new field, click [Create](#).
3. For business context, select [Marketing Program](#).
4. Enter a label for the new field.
An identifier and tooltip are proposed but you can change them. The identifier is the name of the component in the database. This value is enhanced by a specific prefix and suffix, the values shown in front and behind the input field.
5. Select the type of field.
6. If required, enter a field length or values for a field of type list.
7. Click [Create and Edit](#).
8. Under [UIs and Reports](#), you have to enable the field usage for the respective UI or report.
9. To make the new field available, click [Publish](#).

Changing and Deleting an Extension

1. You can change labels and tooltips, and enable or disable custom fields.
2. To save your changes, click [Publish](#).
3. You can also delete custom fields but keep in mind that you have to remove all references to these fields first.

Bringing the Custom Fields to the UI

Once you have created the custom fields, you can bring them to the [Details](#) view by adapting the [Programs](#) application at runtime. For more information, see [Adapting the UI of an SAP Fiori App at Runtime](#) on SAP Help Portal at <http://help.sap.com> ► [Technology Platform](#) ► [SAP NetWeaver](#) ► [User interface add-on 2.0 for SAP NetWeaver](#) ► [Application Help](#) ► [English](#) ► [SAP Fiori Launchpad](#) ► [Using the Launchpad](#) ► [Running an SAP Fiori App](#) ►.

6.2 Marketing Spend Management

6.2.1 Adding New Fields to the “My Marketing Spend - Quick Entry” App

Use

You can add new fields to the [My Marketing Spend - Quick Entry](#) (technical name: `hpa.cei.msm.quickentry`) app according to your business needs for different aspects.

Change Overview Page

To extend the quick entry overview page, the following extensibility entities are available on the different software layers. You can extend each of these entities according to your specific business needs:

Table 45:

UI		Backend/ABAP		
View	Extension Point	Design Time: Gateway Entity	Design Time: Extension Include (in DDIC Structure)	Runtime: Data Structure to Be Redefined
AssignProgramDialog.fragment.xml	extQuickEntryAssignProgramColumnCont	MarketingProgram	--	CUAN_S_MKT_PROGRAM_UI But keep in mind that the fields of this data structure can only be used and no fields can be added.
AssignProgramDialog.fragment.xml	extQuickEntryAssignProgramColumnHeader	MarketingProgram	--	CUAN_S_MKT_PROGRAM_UI But keep in mind that the fields of this data structure can only be used and no fields can be added.
Quickentry.view.xml	extQuickEntryHeaderInfo	InitiativeSpendHeader	INCL_EEW_CUAN_MSM_ROOT_D INCL_EEW_CUAN_MSM_ROOT_DT INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_SPEND_HEADER
InitiativeSpendHeaderTable.view.xml	extQuickEntryTableColumnHead	InitiativeSpendHeader	INCL_EEW_CUAN_MSM_ROOT_D INCL_EEW_CUAN_MSM_ROOT_DT INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_SPEND_HEADER
InitiativeSpendHeaderTable.view.xml	extQuickEntryTableColumnCont	InitiativeSpendHeader	INCL_EEW_CUAN_MSM_ROOT_D INCL_EEW_CUAN_MSM_ROOT_DT INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_SPEND_HEADER

Distribute Spend Amount By Your Own Choice

You can use this controller hook to distribute your spend amount by your own choice.

Table 46:

Controller	Hook
hpa.cei.msm.quickentry.InitiativeSpendHeaderTable.controller.js	extHookOnDisaggregateTotal

More Information

For a general description of the extensibility options and procedures of SAP Fiori apps, see ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.

For more information about extension includes, see ► [SAP Fiori for SAP Business Suite](#) ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) ► [Checking the SAP-Enabled Extension Options](#) ► [Extension Includes](#) in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.

6.2.2 Adding New Fields to the “My Marketing Spend - Details” App

Use

You can add new fields to the [My Marketing Spend - Details](#) (technical name: hpa.cei.msm.maintenance) app according to your business needs for different aspects.

Change List and Detail Page

To extend the list and detail view, the following extensibility entities are available on the different software layers. You can extend each of these entities according to your specific business needs:

Table 47:

UI		Backend/ABAP		
View	Extension Point	Design Time: Gateway Entity	Design Time: Extension Include (in DDIC Structure)	Runtime: Data Structure to Be Redefined
AssignProgramDialog.fragment.xml	extMaintenanceAssignProgramColumnCont	MarketingProgram	--	CUAN_S_MKT_PROGRAM_UI But keep in mind that the fields of this data structure can only be used and no fields can be added.
AssignProgramDialog.fragment.xml	extMaintenanceAssignProgramColumnHeader	MarketingProgram	--	CUAN_S_MKT_PROGRAM_UI But keep in mind that the fields of this data structure can only be used and no fields can be added.
InitiativeList.view.xml	extMaintenanceInitiativeListAttr	Initiative	INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_MSM_UI
Spend.view.xml	extMaintenanceSpendHeaderAttr	MarketingSpendHeader	INCL_EEW_CUAN_MSM_ROOT_D INCL_EEW_CUAN_MSM_ROOT_DT	CUAN_S_MKT_SPEND_HEADER_UI
SpendItemTable.view.xml	extMaintenanceSpendItemColumnHeader	MarketingSpendItem	INCL_EEW_CUAN_MSM_ITEM_D INCL_EEW_CUAN_MSM_ITEM_DT	CUAN_S_MKT_SPEND_ITEM_UI
SpendItemTable.view.xml	extMaintenanceSpendItemColumnCont	MarketingSpendItem	INCL_EEW_CUAN_MSM_ITEM_D INCL_EEW_CUAN_MSM_ITEM_DT	CUAN_S_MKT_SPEND_ITEM_UI

Distribute Spend Amount By Your Own Choice

You can use this controller hook to distribute your spend amount by your own choice.

Table 48:

Controller	Hook
hpa.cei.msm.maintenance.SpendItemTable.controller.js	extHookOnDisaggregateTotal

More Information

For a general description of the extensibility options and procedures of SAP Fiori apps, see ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.

For more information about extension includes, see ► [SAP Fiori for SAP Business Suite](#) ► [Extensibility Information for SAP Fiori](#) ► [Extensibility](#) ► [Checking the SAP-Enabled Extension Options](#) ► [Extension Includes](#) in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.

6.3 Calendar

In the standard system, you can use the personalization to configure the user interface for every device you use (desktop, tablet, and phone). You can for example hide standard widgets or define the order in which they are displayed on an individual basis.

You can use the extensions described in the following chapters to adapt the *Marketing Calendar* app to meet your business requirements. You can do the following:

- Use customer-defined texts (for example terminology). For more information, see [Customer-Defined Texts \[page 157\]](#).
- Add customer-defined fields and/or customer-defined filter criteria to the standard marketing calendar. For more information, see [Customer-Defined Fields and Filter Criteria \[page 159\]](#).
- Create customer-defined widgets and make these available in the marketing calendar. You can use the personalization to display customer-defined widgets when you configure the user interface. For more information, see [Customer-Defined Widgets \[page 166\]](#).
- Define a customer-defined user interface, which every user can adjust to meet their requirements using the personalization. The customer-defined user interface can contain standard widgets and customer-defined widgets. It replaces the standard SAP interface. For more information, see [Customer-Defined Configuration \[page 174\]](#).
- Deactivate personalization; user-defined configuration of the user interface is then no longer possible. For more information, see [Deactivate Personalization \[page 177\]](#).

i Note

The standard app *My Marketing Calendar* (technical name: hpa.cei.mkt.cal) consumes its own gateway service (OData Service) with the external name CUAN_MKTCAL_SRV. This is the OData service that is described in this text. OData entity is an entity in this OData Service (unless another is explicitly mentioned).

For references to the standard SAP documentation, see [More Information \[page 178\]](#). This documents describe the tools and technologies that are available to develop customer-defined extensions. The extension options for the calendar are described in the following chapters.

6.3.1 Customer-Defined Texts

Use

You can replace the texts (such as terminology) used in the standard system with customer-defined texts. To do so, you must implement the following **controller extension**:

Table 49:

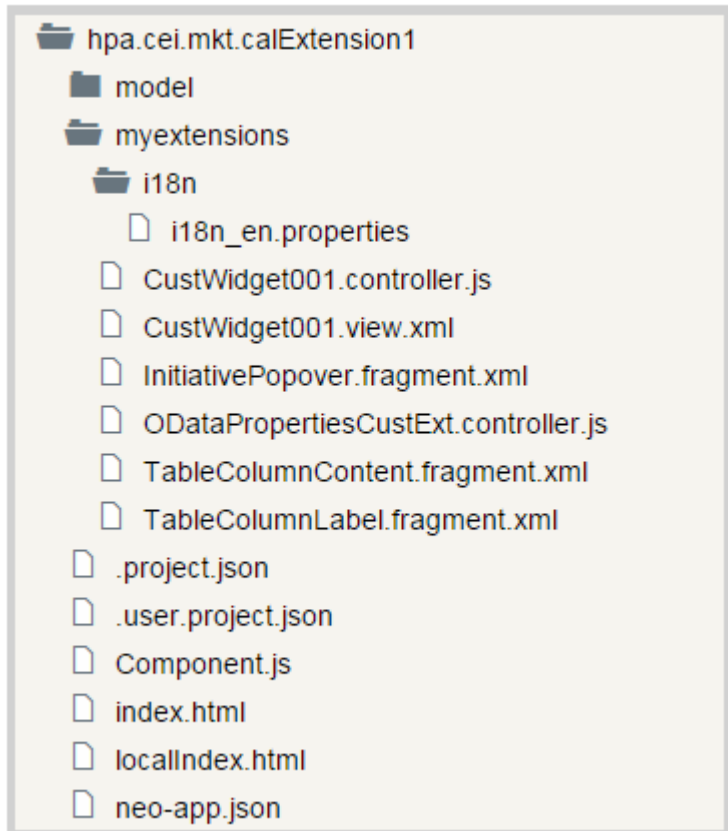
Controller	Hook
hpa.cei.mkt.cal.Component	extHookGetExtensionBundleName

In this hook, you define the path for (language-dependent) text files, from which the system is to obtain the customer-defined texts. Record the customer-defined texts that are to be displayed instead of the standard texts in these text files.

If the system does not find any customer-defined texts for a key, it displays the standard texts instead.

Example

The extension options are explained using examples. The example is based on the extension project **hpa.cei.mkt.calExtension** with the following structure:



Customer-Defined Texts

File: myextensions.i18n.i18n_en.properties

```
#XTIT,50: Title for the fullscreen section FULLSCREEN_TITLE=My <<< extended >>>
Marketing Calendar #XTIT,50: Title for the campaign list control
TitleCampaignList=<<< extended >>> Campaign List
```

Enhancements in file Component.js (of extension project)

```
this.hpa.cei.mkt.cal.Component.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.Com
ponent", { metadata: { }, // metadata // hook
hpa.cei.mkt.cal.Component~extHookGetExtensionBundleName
extHookGetExtensionBundleName: function(){ var sExtBundleName =
"hpa.cei.mkt.calExtension1.myextensions.i18n.i18n"; return
sExtBundleName; } // extHookGetExtensionBundleName });
```

6.3.2 Customer-Defined Fields and Filter Criteria

Use

You can do the following to the fields and filters:

- Add customer-defined fields to the campaign view; the fields can be
 - Shown in the [Calendar View](#) (on the popover)
 - Shown in the [List View](#)
 - Included in the function [Export to Spreadsheet](#) which is available in the [List View](#)
- Change the existing filter options used to search for campaigns, by adding new filter criteria or deactivating standard filter criteria.

Customer-defined fields and filter criteria must be available (meaning that they are explicitly defined or automatically added) as a property for the (OData) entity [Initiative](#) in the OData service consumed. Customer-defined filter criteria must also be available in the (OData) entity for which they are to be used (such as [InitiativeAnalysis](#)).

Caution

The extension concept makes it possible to add customer-defined fields and filters, which are to be displayed in the campaign view ([Calendar View](#), [List View](#)) and can be used as a storage level for standard key figures. The extension concept does not allow you to use (display) customer-defined key figures in the standard widgets. If you want to do this, you must add customer-defined widgets.

It is also not possible to use customer-defined OData entities in the standard widgets.

Note

You must define the extension points and controller extensions to be used for your customer extension. You do this by making the corresponding entries in the file `Component.js`. For more information, see section [Example](#).

Add Customer-Defined Fields

UI Extension Points

If you want to visualize customer-defined fields or standard fields that were not previously visualized in the Calendar, you have to use the following **extension points** and extend both the corresponding databases (incl. HANA views), structures, and the OData entities:

Table 50:

View	Extension point	Design time: gateway entity	Design time extension include	Runtime
CampaignList.view.xml	MktCalCampaignList-TableColumnLabel	Initiative	INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_UI
	MktCalCampaignList-TableColumnContent	Initiative	INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_UI

View	Extension point	Design time: gateway entity	Design time extension include	Runtime
MarketingGanttChartInitiativePopover.fragment.xml	MktCalGantChartInitiativePopover	Initiative	INCL_EEW_CUAN_INITIATIVE_ROOT	CUAN_S_INITIATIVE_UI

The extension points mentioned above contain the standard SAP implementation. By using these extension points, you can add customer-defined fields and remove fields shown in the standard SAP layout. If you extend the view [CampaignList.view.xml](#), you must ensure that both extension points remain in synch:

`MktCalCampaignListTableColumnLabel` defines the header labels of the table,

`MktCalCampaignListTableColumnContent` defines the table rows containing the field values.

Provide Fields in ABAP Backend System

If you want to visualize customer-defined fields in the Calendar, you have to ensure that these fields are provided by the ABAP backend system.

Remarks about extending the data basis in the ABAP backend system:

- If you want to add customer-defined fields (that are not available in the standard SAP system) you have to add these fields to the extension include `INCL_EEW_CUAN_INITIATIVE_ROOT`. By doing this, the new field is automatically:
 - Added to data base (table `CUAN_D_INI_ROOT`)
 - Included in BOPF objects
 - Available as a property for the OData entity `Initiative`
- If you want to add fields that are already available as a property for the OData entity `Initiative` (and are therefore part of the structure `CUAN_S_INITIATIVE_UI`) but not yet requested by the app [My Marketing Calendar](#), you do **not** need to change the extension include or the OData Service.
- If you want to add fields that are part of the structure `CUAN_S_INITIATIVE_UI` (and are therefore already available in the backend system), but not yet defined as a property for the OData entity `Initiative`, you have to create a customer-defined OData service and manually add the fields to the property list for the OData entity `Initiative`. To create a customer-defined OData service, you have to create a new OData service (in the customer namespace) that redefines the standard OData service. In the new OData service created, you only have to adjust the property list for OData entity `Initiative`. You also have to extend the structure `CUAN_S_QR_INI_W_MAIN_IA`.

Note: In this case you cannot include the field in the extension include `INCL_EEW_CUAN_INITIATIVE_ROOT` because the field is already part of the structure `CUAN_S_INITIATIVE_UI`.

Remarks about enhancing SAP HANA information models:

- If you want to add new fields to the Calendar, you have to ensure that the fields are available in the corresponding HANA views. If the new field is not found in the standard HANA view `sap.hana-app.cuan.common.internal.AT_SEARCH_INITIATIVE`, use the mass copy function in SAP HANA Studio to copy and enhance the HANA views.
- Once you have executed the mass copy function, you must delete the buffer memory in the BOPF in Customizing for [SAP hybris Marketing](#) under **General Settings** > **Clear BOPF Metadata from Buffer**.

UI Controller Extension

If you want to visualize customer-defined fields in the Calendar, you have to use the following **controller extensions**:

Table 51:

Controller	Hook
hpa.cei.mkt.cal.view.CampaignList	extHookProvideODataPropertiesInitiatives
hpa.cei.mkt.cal.view.MarketingGanttChart	extHookProvideODataPropertiesInitiatives
hpa.cei.mkt.cal.view.IniCategoriesAndChannels	extHookProvideODataPropertiesInitiatives
hpa.cei.mkt.cal.view.CampaignList	extHookAdditionalFieldsForDataExport

By implementing the hook `extHookProvideODataPropertiesInitiatives`, you can add customer-defined fields (and fields that have not previously been requested) to the list of fields that are requested from the OData service in the standard SAP system. Alternatively, you can also create a completely new list of fields to be requested in the hook implementation, enter *Add* or *Replace* in the return parameter `Indicator` `AddOrReplace`. Check that the fields are not requested more than once.

Remarks:

- Not every field provided by the OData Service is requested from the app *My Marketing Calendar*
- The list of fields (for OData entity *Initiative*) requested from the OData Service in the standard SAP system can be seen in `hpa.cei.mkt.cal.util.ODataProperties.js`.
- You must only implement the hook `extHookProvideODataPropertiesInitiatives` once, but should assign the controllers described above to it (in the file `component.js`), to avoid unnecessary OData calls.

By implementing the hook `extHookAdditionalFieldsForDataExport`, you can include customer-defined fields and standard fields that were not previously observed in the function **Export to Spreadsheet**.


Add Customer-Defined Filters

Provide Field (to be used as filter) in ABAP Backend System

If you want to use customer-defined filters, you have to provide the corresponding fields in the data sources and in the OData service; this can be done as described in section *Add Customer-Defined Fields*. If you want to use additional standard fields as a filter only, some of the described steps may not be necessary.

Remarks about extending SAP HANA Information Models and a data basis and OData service in the ABAP backend system:

- You have to ensure that the fields to be used as a filter are available in the following HANA views:
 - `sap.hana-app.cuan.common.internal.AT_SEARCH_INITIATIVE`
 - `sap.hana-app.cuan.mktcal.internal.CA_MKTCAL_FILTER`
 - `sap.hana-app.cuan.mktcal.internal.CA_MKTCAL_INI_AVG_ACCESS`
 - `sap.hana-app.cuan.mktcal.internal.CA_MKTCAL_INTEREST_SENTIMENTS`
 - `sap.hana-app.cuan.mktcal.internal.CA_MKTCAL_TOP_SPENDS`
 - `sap.hana-app.cuan.mktcal.internal.CA_MKTCAL_SPEND_BY_PROGRAM`

If the customer-defined field is not included in the HANA views, use the mass copy function in SAP HANA Studio to copy and enhance HANA Views. Having done so, execute the Customizing for *SAP hybris Marketing* under **General Settings** > *Clear BOPF Metadata from Buffer* .

- You have to ensure that customer-defined filters are available as properties in the following OData entities:
 - Initiative
 - InitiativeAnalysis

Depending on the fields to be used as customer-defined filter criteria, you have to extend some of the following structures:

- INCL_EEW_CUAN_INITIATIVE_ROOT (customer-defined fields, enhancement for OData entity `Initiative`): Field is automatically added as a property for the entity `Initiative`.
- CUAN_S_QR_INI_W_MAIN_IA (standard fields not defined as a property for the OData entity `Initiative`; enhancement for OData entity `Initiative`): Field must be manually added as a property for the entity `Initiative`.
- INCL_EEW_CUAN_MKTCAL_FILTER (additional fields (used as a filter) for other OData entities): Field is automatically added as a property for other OData entities.

Business Add-Ins (ABAP backend system)

To make customer-defined filter criteria (and filter values) visible on the popup [Filter](#) in the app [My Marketing Calendar](#) you have to implement the Business Add-In `BADI_CUAN_MKTCAL_FILTER` in the enhancement spot `ES_CUAN_MARKETING_CALENDAR`. For more information about implementing the BAdI, see the documentation in [Customizing under SAP hybris Marketing](#) ► [Campaign](#) ► [Calendar](#) ► [BAdI: Definition of Filter Properties and Filter Values](#) ►.

UI Controller Extension

If you want to use customer-defined filter criteria in the app [My Marketing Calendar](#), you have to use the following **controller extension**:

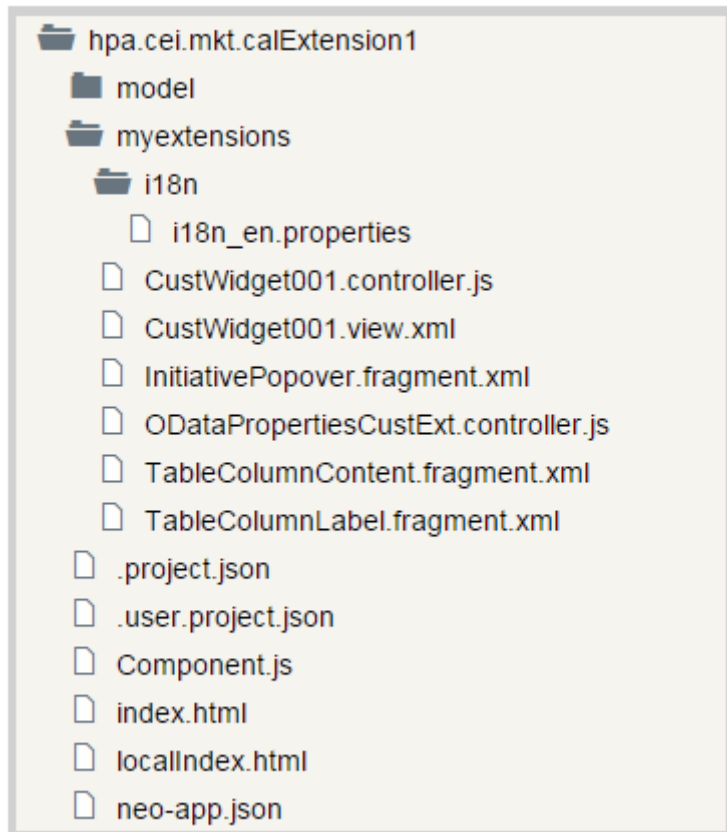
Table 52:

Controller	Hook
<code>hpa.cei.mkt.cal.Component</code>	<code>extHookOnFilterManagerInstantiated</code>

By implementing the hook `extHookOnFilterManagerInstantiated` you can add customer-defined filter criteria to the filter manager (this is a coding part of the SAP Fiori app that manages filters and filter values). You must implement this hook in addition to the Business Add-In to ensure that filter criteria are also applied when searching for campaigns.

Example

The extension options available are explained using examples. The example is based on the extension project **hpa.cei.mkt.calExtension** with the following structure:



Hook `extHookProvideODataPropertiesInitiatives`

File: `myextensions.ODataPropertiesCustExt.controller.js`

```
sap.ui.controller("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.ODataPropertiesCustExt", {
  extHookProvideODataPropertiesInitiatives:
  function(aODataPropertiesSAP) {
    return {
      sIndicatorAddOrReplace: "Add",
      aODataPropertiesCust: ["ZzSeasonId"]
    };
  }, //
  extHookProvideODataPropertiesCust
});
```

Hook `extHookAdditionalFieldsForDataExport`

File: `myextensions.ODataPropertiesCustExt.controller.js`

```
sap.ui.controller("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.ODataPropertiesCustExt", {
  extHookAdditionalFieldsForDataExport: function(oExport) {
    var sName; var oTemplate; var oNewColumn;
    sName = "ZZ Season"; oTemplate =
    {
      content: "{ZzSeasonId}"
    }; oNewColumn = new
    sap.ui.core.util.ExportColumn("", {
      name: sName,
      template:
      oTemplate
    }); oExport.addColumn(oNewColumn);
  }, //
  extHookAdditionalFieldsForDataExport
});
```

Extension Point `MktCalCampaignListTableColumnContent`

File: myextensions.TableColumnContent.fragment.xml

```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
<ObjectIdentifier title="{Name}" text="{InitiativeIdExt}"
class="sapMTableContentMargin" /> <Text text="{ path:'StartDate',
type:'sap.ca.ui.model.type.Date', formatOptions: {style:'daysAgo'}} -
{ path:'EndDate', type:'sap.ca.ui.model.type.Date', formatOptions:
{style:'daysAgo'}}" /> <Text text="{Category/CategoryDescription}" /> <Text
text="{Priority/PriorityDescription}" /> <Text text="{LifecycleStatus/
StatusDescription}" /> <Text text="{MarketingAreaDescription}" /> <!-- ***
remove column **** --> <!-- <Text text="{ProgramName}" /> --> <Text
text="{ parts:[ {path: 'MainInteractionProgress/Progress'}, {path:
'MainInteractionProgress/Tooltip'} ],
formatter:'hpa.cei.mkt.cal.util.Formatter.getInitiativeSuccessText' }"/> <!-- add
column --> <Text text="{InitiativeId}" /> <Text text="{ZzSeasonId}" /> </
core:FragmentDefinition>
```

Extension Point MktCalCampaignListTableColumnLabel

File: myextensions.TableColumnLabel.fragment.xml

```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core">
<Column> <Text text="{i18n>LabelInitiative}" /> </Column> <Column
minScreenWidth="Tablet" demandPopin="true"> <Text
text="{i18n>LabelDateStartEnd}" /> </Column> <Column minScreenWidth="Tablet"
demandPopin="true"> <Text text="{i18n>LabelCategory}" /> </Column>
<Column minScreenWidth="Tablet" demandPopin="true"> <Text
text="{i18n>LabelPriority}" /> </Column> <Column minScreenWidth="Tablet"
demandPopin="true"> <Text text="{i18n>LabelStatus}" /> </Column>
<Column minScreenWidth="Tablet" demandPopin="true"> <Text
text="{i18n>LabelMarketingArea}" /> </Column> <!-- remove this column
<Column minScreenWidth="Tablet" demandPopin="true"> <Text
text="{i18n>LabelProgram}" /> </Column> --> <Column minScreenWidth="Tablet"
demandPopin="true"> <Text text="{i18n>TitleIconTabBarCampaignSuccess}" /> </
Column> <!-- new column --> <Column minScreenWidth="Tablet"
demandPopin="true"> <Text text="{i18n>LabelInitiativeId}" /> </Column>
<Column minScreenWidth="Tablet" demandPopin="true"> <Text text="ZZ Season"/
> </Column> </core:FragmentDefinition>
```

Extension Point MktCalGantChartInitiativePopover

File: myextensions.InitiativePopover.fragment.xml

```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"
xmlns:l="sap.ui.layout" xmlns:f="sap.ui.layout.form"> <f:SimpleForm
id="idInitiativePopoverGeneralForm" editable="false"
layout="ResponsiveGridLayout" labelSpanL="2" labelSpanM="2" labelSpanS="6"
emptySpanL="0" emptySpanM="0" emptySpanS="0" columnsL="2" columnsM="2" columnsS="2"
> <f:content> <Label text="{i18n>LabelInitiative}" /> <Text
text="{PopoverInitiative>InitiativeIdExt}" /> <Label
text="{i18n>LabelStartDate}" /> <Text
text="{path:'PopoverInitiative>StartDate', type:'sap.ca.ui.model.type.Date',
formatOptions: {style:'daysAgo'}}" /> <Label text="{i18n>LabelEndDate}" /
> <Text text="{path:'PopoverInitiative>EndDate',
type:'sap.ca.ui.model.type.Date', formatOptions: {style:'daysAgo'}}" />
<Label text="{i18n>LabelDescription}" /> <Text
text="{PopoverInitiative>Description}" /> <Label
text="{i18n>LabelCategory}" /> <Text text="{PopoverInitiative>Category/
CategoryDescription}" /> <Label text="{i18n>LabelStatus}" /> <Text
text="{PopoverInitiative>LifecycleStatus/StatusDescription}" /> <Label
text="{i18n>LabelPriority}" /> <Text text="{PopoverInitiative>Priority/
PriorityDescription}" /> <Label text="{i18n>LabelOwner}" /> <Text
text="{PopoverInitiative>MainResponsiblePersonName}" /> <Label
text="{i18n>LabelMarketingArea}" /> <Text
text="{PopoverInitiative>MarketingAreaDescription}" /> <!-- new line -->
<Label text="!!!!!" /> <Text text="Extension Begin: Program removed, some
```



```

fields added"/>          <Label text="{i18n>LabelInitiativeId}" />          <Text
text="{PopoverInitiative>InitiativeId}"/>          <Label text="Season" />
<Text text="{PopoverInitiative>ZzSeasonId}"/> <!-- do not show the program
name          <Label text="{i18n>LabelProgram}" />          <Text
text="{PopoverInitiative>ProgramName}"/> -->          <Label text="!!!!!" /
>          <Text text="Extension End"/>          </f:content>          </f:SimpleForm>
<f:SimpleForm id="idInitiativePopoverSuccessForm" editable="false"
layout="ResponsiveGridLayout"          labelSpanL="2" labelSpanM="2" labelSpanS="6"
emptySpanL="0" emptySpanM="0" emptySpanS="0" columnsL="2" columnsM="2" columnsS="2"
>
  <f:content>          <Label
text="{i18n>TitleIconTabBarCampaignSuccess}" />          <Text text="{parts:
[
  {path: 'PopoverInitiative>MainInteractionProgress/Progress'},
{path: 'PopoverInitiative>MainInteractionProgress/Tooltip'}]],
formatter: 'hpa.cei.mkt.cal.util.Formatter.getInitiativeSuccessText'}"/>          </
f:content>          </f:SimpleForm>          <f:SimpleForm
id="idInitiativePopoverSpendForm" editable="false"
layout="ResponsiveGridLayout"          labelSpanL="2" labelSpanM="2" labelSpanS="6"
emptySpanL="0" emptySpanM="0" emptySpanS="0" columnsL="2" columnsM="2" columnsS="2"
>
  <f:content>          </f:content>          </f:SimpleForm>          <f:SimpleForm
id="idInitiativePopoverTargetGroupForm" editable="false"
layout="ResponsiveGridLayout"          labelSpanL="2" labelSpanM="2" labelSpanS="6"
emptySpanL="0" emptySpanM="0" emptySpanS="0" columnsL="2" columnsM="2" columnsS="2"
>
  <f:content>          </f:content>          </f:SimpleForm>          <f:SimpleForm
id="idInitiativePopoverInterestsForm" editable="false"
layout="ResponsiveGridLayout"          labelSpanL="2" labelSpanM="2" labelSpanS="6"
emptySpanL="0" emptySpanM="0" emptySpanS="0" columnsL="2" columnsM="2" columnsS="2"
>
  <f:content>          </f:content>          </f:SimpleForm>          <f:SimpleForm
id="idInitiativePopoverNavigationForm" >          <f:content>          <Link id
="idInitiativePopoverNavigationLink" text="{i18n>NAVLINK_CAMPAIGN}"
visible="{PopoverInitiative>IsLinkVisible}" press="onHandleNavigation"/>          </
f:content>          </f:SimpleForm> </core:FragmentDefinition>

```

Extensions to the File Component.js (in the Extension Project)

```

this.hpa.cei.mkt.cal.Component.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.Com
ponent", { metadata: { version: "1.0", // if you need to redefine
the ODATA service: propagate the new service config: {
"sap.ca.serviceConfigs": [{ name: "CUAN_MKTCAL_SRV", serviceUrl:
"/sap/opu/odata/sap/ZCUAN_MKTCAL_EXT_1_SRV/" } ] }, //
config // add customizing for hooks and extension points (assigned to a
controller) customizing: { "sap.ui.viewExtensions": {
"hpa.cei.mkt.cal.view.CampaignList": {
"MktCalCampaignListTableColumnLabel": { className:
"sap.ui.core.Fragment", fragmentName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.TableColumnLabel",
type: "XML" }, //
MktCalCampaignListTableColumnLabel
"MktCalCampaignListTableColumnContent": { className:
"sap.ui.core.Fragment", fragmentName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.TableColumnContent",
type: "XML" } //
MktCalCampaignListTableColumnContent }, // "...CampaignList"
"hpa.cei.mkt.cal.view.MarketingGanttChartInitiativePopover": {
"MktCalGantChartInitiativePopover": { className:
"sap.ui.core.Fragment", fragmentName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.InitiativePopover",
type: "XML" } //
MktCalGantChartInitiativePopover } // "...InitiativePopover" }, //
sap.ui.viewExtensions "sap.ui.controllerExtensions": {
"hpa.cei.mkt.cal.view.CampaignList": { controllerName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.ODataPropertiesCustExt"
}, // "hpa.cei.mkt.cal.view.CampaignList"
"hpa.cei.mkt.cal.view.MarketingGanttChart": { controllerName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.ODataPropertiesCustExt"
}, // "hpa.cei.mkt.cal.view.CampaignList"
"hpa.cei.mkt.cal.view.IniCategoriesAndChannels": { controllerName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.ODataPropertiesCustExt"

```

```

    } // "hpa.cei.mkt.cal.view.CampaignList"    } //
sap.ui.controllerExtensions    } // customizing    }, // metadata    // hook
hpa.cei.mkt.cal.Component~extHookOnFilterManagerInstantiated
extHookOnFilterManagerInstantiated: function() {    var
oFilterManagementExtensions = {    abstractFilterProperties: [{
filterPropertyConstant: "CO_ABS_PROP_INITIATIVE_SEASON",
filterPropertyName: "ABS_Initiative_Season"    },    {
filterPropertyConstant: "CO_ABS_PROP_INITIATIVE_ID",    filterPropertyName:
"ABS_Initiative_Id"    }    ],
odataEntityFilterPropertyNames: [{    odataEntity: "Initiative",
filterPropertyName: "ABS_Initiative_Season",
odataEntityFilterPropertyName: "ZzSeasonId"    },    {
odataEntity: "InitiativeAnalysis",    filterPropertyName:
"ABS_Initiative_Season",    odataEntityFilterPropertyName:
"ZzSeasonId"    },    {    odataEntity:
"Initiative",    filterPropertyName: "ABS_Initiative_Id",
odataEntityFilterPropertyName: "InitiativeId"    },    {
odataEntity: "InitiativeAnalysis",    filterPropertyName:
"ABS_Initiative_Id",    odataEntityFilterPropertyName:
"InitiativeId"    }    ]    };    return
oFilterManagementExtensions;    }, // extHookOnFilterManagerInstantiated

```

6.3.3 Customer-Defined Widgets

You can program customer-defined widgets and include these in the Calendar. To do so, you must program the actual widgets and use a **controller extension** to make these known to the Calendar.

Table 53:

Controller	Hook
hpa.cei.mkt.cal.Component	extHookGetCustomerWidgets

In this hook, you define the metadata for the customer-defined widgets and in doing so insert the widgets in the list of widgets that can be displayed in the Calendar. The coding for the actual widget is found in a separate file and is not part of the hook implementation.

The metadata for a widget includes the following information:

Table 54:

Information/Parameter	Data Type	Note
id	String	Internal identification for widgets; must be defined uniquely for all widgets (SAP, customer defined)
name	String	
description	String	
icon	String	Is displayed in the personalization and helps users to configure the layout.
expandable	Boolean	
		Defines whether you can make the widget bigger (true) or not (false)

Information/Parameter	Data Type	Note
size	String	Defines the size of the widget (large = entire width, small = half width)
viewName	String	The name of the view in which the implementation of the widget is found consists of the name and the type of view.
viewType	String	
showHeader	Boolean	Defines whether the widget is displayed in a container (true) or not (false). The container contains the name of the widget and the expandable button if necessary)

Users of the calendar can use the personalization to display the customer-defined widgets in the Calendar.

i Note

The app automatically recognizes which widget (including customer-defined widgets) is currently active and manages this information as part of the AppState (for example when the [Save as Tile](#) function is executed). However, the system is not able to automatically identify the status in customer-defined widgets, meaning it cannot manage this in the AppState. If this is required, this must be programmed in the customer-defined widget. For examples of this, see the standard widgets listed under [Customer-Defined Configuration \[page 174\]](#)

Data Retrieval in Customer-Defined Widgets

The following options are available for making the data to be displayed available in customer-defined widgets:

- Use the OData Models available in the Calendar
The OData service (CUAN_MKTCAL_SRV) available in the standard system is used for data retrieval. This includes creating a customer-defined OData service that redefines the standard OData service and adds customer-defined fields or entities to it.

The following alternatives are available for data retrieval:

- Read directly
In the controller for the customer-defined widget, the OData model used in the standard system is first determined, for example by the following statements:

```
var oModel = sap.ca.scfld.md.app.Application.getImpl().getODataModel();
```

The data is then read. The existing options for doing so are described in the SAPUI5 documentation. For more information, see [More Information \[page 178\]](#).

- Read data using the ODataBatchHandler available in the standard system (see section [API Documentation ODataBatchHandler](#))
By using the ODataBatchHandler, you can ensure that the same data (same entity and same field list) is not requested more than once and can perform calls to the back-end system in parallel. The following functions must be programmed in the controller for the customer-defined widget (see also the examples in the section [Example > Actual Customer-Defined Widget](#)).

- Registration of the widget / view in the ODataBatchHandler
- Deregistration in the ODataBatchHandler
- Implementation of the callback functions (getODataRequest, onODataReadSuccess) for ODataBatchHandler
 - Parameters for the OData call are collected in the function getODataRequest (entity, list of fields to be selected, filter criteria, sort criteria). The function is called automatically when the app is started and if the app criteria are changed.
 - Data processing (such as visualization) is programmed in the function onODataReadSuccess.

- Use a different OData model

Other OData models are to be used for data retrieval. This includes other OData services found in the standard system (not therefore CUAN_MKTCAL_SRV) and any OData services that are created by customers.

The following alternatives are available for data retrieval:

- Read directly

In the controller for the customer-defined widget, a new OData model is first created, for example by the following statement:

```
var oModel = new sap.ui.model.odata.ODataModel("/", { json : true });
```

The data is then read. The existing options for doing so are described in the SAPUI5 documentation.

- Read data using an additional instance of the ODataBatchHandler available in the standard system (see the section *API Documentation ODataBatchHandler*)

This alternative is recommended if data from the same OData service is to be requested from various customer-defined widgets (parallel processing, avoid identical calls).

The following functions must be programmed in the controller for the customer-defined widget:

- Generate an instance of the ODataBatchHandler and attach it to the component controller (to avoid redundant data).

```
If (!this.getOwnerComponent().myOwnBatchHandler) {
  this.getOwnerComponent().myOwnBatchHandler = new
  hpa.cei.mkt.cal.util.ODataBatchHandler(...);
}
```

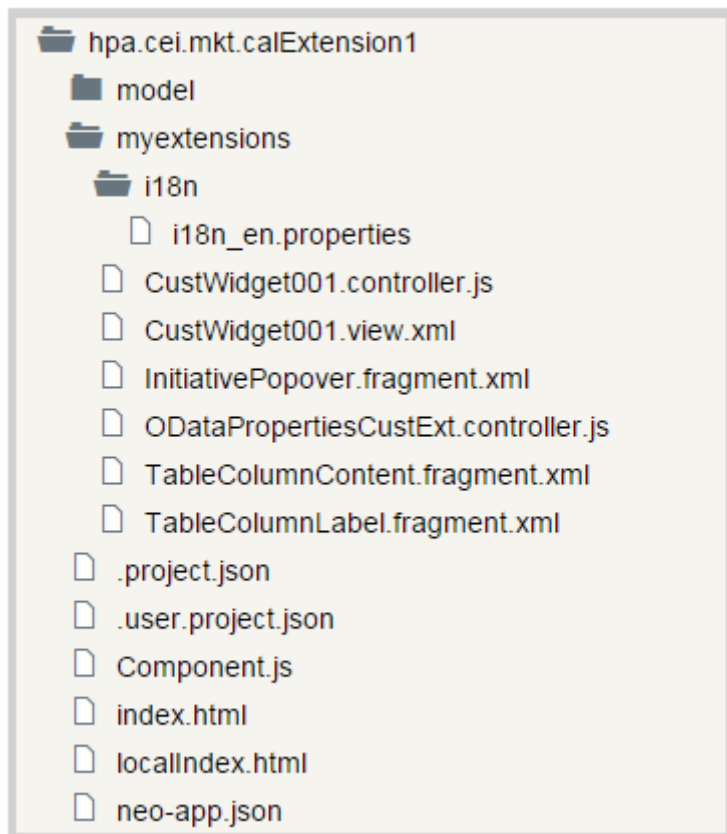
- Program the following functions for use of the ODataBatchHandler (similar to ► [Use the OData models available in the Calendar](#) ► [Read data using the ODataBatchHandler available in the standard system](#) ►).
- Registration of the widget / view in the ODataBatchHandler
- Deregistration in the ODataBatchHandler
- Implementation of the callback functions (getODataRequest, onODataReadSuccess) for the ODataBatchHandler

- Use other models

It is also possible to access data from other data sources without having to use an OData service. Access to this data must be programmed customer-defined and cannot be described here.

Example

The extension options are explained using examples. The examples are based on the extension project **hpa.cei.mkt.calExtension** with the following structure:



Actual Customer-Defined Widget

File: myextensions.CustWidget001.controller.js

```
sap.ui.core.mvc.Controller.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.CustWidget001", { // registration of the widget with ODataBatchHandler,
existing ODataModel is re-used  onInit: function() {
this.getOwnerComponent().BatchHandler.registerView(this, this.getODataRequest,
this.onODataReadSuccess); }, // onInit // de-registration of the widget with
ODataBatchHandler, existing ODataModel is re-used  onExit: function() {
this.getOwnerComponent().BatchHandler.unregisterView(this, this.getODataRequest,
this.onODataReadSuccess); }, // onExit // implementation of callback function
getODataRequest (ODataBatchHandler), existing ODataModel is re-used
getODataRequest: function() { // return Entity, Field list, Filters and
Sorters (see also implementation for standard widget) }, // getODataRequest //
implementation for callback function onODataReadSuccess (ODataBatchHandler),
existing ODataModel is re-used  onODataReadSuccess: function(oJsonData) { //
define how data shall be processed }, // onODataReadSuccess });
```

i Note

When the controller for a customer-defined widget is programmed, you can use the base controller `hpa.cei.mkt.cal.util.BaseController` supplied. This defines functions that are used to provide a simple access to frequently used objects (such as `getBatchHandler` or `getEventBus`).

You can also implement two methods in the controller for a customer-defined widget, which are accessed by the Marketing Calendar Framework. These functions provide the implementation of the widget with additional information:

Table 55:

Function in Controller	Parameter	Meaning
<code>setMetadata</code>	JSON-Objekt with the attributes <code>widgetMetadata</code> and <code>instanceCounter</code>	Access to the meta data in the widget and the sequence number of the widget instance (a widget can be included more than once in the personalization). Is called once when the widget is generated.
<code>onWidgetExpandToggle</code>	Two parameters: <code>bPressed</code> and <code>sWidgetHeight</code>	Notification if the user extends the widget. During expansion, the first parameter has the value <code>true</code> . When it is reduced, it has the value <code>false</code> . The second parameter specifies the new height of the widget. It is always called if the user has expanded or reduced the size of the widget.

File: `myextensions.CustWidget001.view.xml`

```
<core:View xmlns:core="sap.ui.core" xmlns:mvc="sap.ui.core.mvc"
xmlns:l="sap.ui.layout" xmlns="sap.m"
xmlns:suite="sap.suite.ui.commons"
controllerName="hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.CustWidget001"
xmlns:html="http://www.w3.org/1999/xhtml">
  <HBox
    alignItems="Center" justifyContent="Center">
    <VBox alignItems="Center"
    justifyContent="Center">
      <FlexBox height="1rem"/>
      <Text
        text="Very very important Information!!!" />
      <FlexBox height="2rem"/>
    </VBox>
    <HBox alignItems="Center" justifyContent="Center">
      <core:Icon src="sap-icon://puzzle" size="2rem" />
      <FlexBox
        width="3rem"/>
      <Label design="Bold" text="Be prepared -- X-Mas is
        coming soon!!!" />
      <FlexBox width="3rem"/>
    </HBox>
  </HBox>
</core:View>
```

Extensions to the File `Component.js` (in the Extension Project)

```
this.hpa.cei.mkt.cal.Component.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.Com
ponent", {
  metadata: { // existing ODataModel is re-used, OData service is
    redefined (new service: ZCUAN_MKTCAL_EXT_1_SRV)
    config: {
      "sap.ca.serviceConfigs": [{
        name: "CUAN_MKTCAL_SRV",
        serviceUrl:
        "/sap/opu/odata/sap/ZCUAN_MKTCAL_EXT_1_SRV/"
      }], // config
    }, //
    metadata // hook hpa.cei.mkt.cal.Component~extHookGetCustomerWidgets
    extHookGetCustomerWidgets: function() {
      return ({
        CustWidget001:
        {
          id: "CustWidget001",
          name: "CustWidget001***Hello World",

```

```

description: "This is Custom Widget 001",           expandable: true,           size:
"small",           viewName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.CustWidget001",
viewType: "XML",           icon: "sap-icon://physical-activity"           },
CustWidget002: {           id: "CustWidget002",           name: "CustWidget002***Hello
World",           description: "This is Custom Widget 002",           expandable:
false,           size: "large",           viewName:
"hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.myextensions.CustWidget002",
viewType: "XML",           icon: "sap-icon://idea-wall"           }           }); //
extHookGetCustomerWidgets });

```

API Documentation OdataBatchHandler (Class hpa.cei.mkt.cal.util.OdataBatchHandler)

Extends sap.ui.base.ManagedObject

Defined in: ODataBatchHandler.js.

Table 56:

Class Summary	
	hpa.cei.mkt.cal.util.ODataBatchHandler (sId, mSettings) Constructs and initializes an OData batch handler instance with the given sId and mSettings.

Table 57:

Method Summary	
	registerView (oController, fnRequestPayloadCb, fnSuccessCb) Register a view controller, its request payload and success callback functions in the OData batch handler.
	triggerODataRead () Trigger an OData read call for all registered views.
	unregisterView (oController, fnRequestPayloadCb, fnSuccessCb) The unregister function for the registerView() function.

Table 58:

Class Detail
<p>hpa.cei.mkt.cal.util.ODataBatchHandler(sId, mSettings)</p> <p>Constructs and initializes an OData batch handler instance with the given <code>sId</code> and <code>mSettings</code>. This class is used to collect OData GET requests for a specific OData service from various registered controls, execute the requests in one \$batch call and dispatch the OData responses to the corresponding controls.</p> <p>The JSON object <code>mSettings</code> is required and must contain the mandatory property:</p> <ul style="list-style-type: none"> • <code>EventBus</code> with a valid <code>{sap.ui.core.EventBus}</code> object of the current application. • <code>TextBundle</code> with a valid text resource bundle of the current application for error message texts <p>The <code>ODataBatchHandler</code> requires a valid OData model <code>{sap.ui.model.odata.ODataModel}</code>. The model can be set:</p> <ul style="list-style-type: none"> • During instantiation by passing it using <code>mSettings.ODataModel</code>. • After instantiation using the <code>setODataModel()</code> method. <p>Controls that want to use the functionality must register themselves in the batch handler using the <code>registerView()</code> function and implement the required callback functions.</p> <p>The OData batch handler is also responsible for setting and releasing the SAP Fiori busy indicator. If errors occur, dialog boxes are shown with messages referencing the affected controls.</p> <p>Parameters:</p> <p><i>{string} sId Optional</i></p> <p>Id for the new object, generated automatically if no id is given.</p> <p><i>{object} mSettings</i></p> <p>Initial settings for the new control.</p>

Table 59:

Method Detail
<p>registerView(oController, fnRequestPayloadCb, fnSuccessCb)</p> <p>Register a view controller, its request payload and success callback functions in the OData batch handler. When the functions are called, the context of the callback functions is bound to <code>oController</code>.</p> <p>The function <code>fnRequestPayloadCb</code> is called for each registered view whenever the filter criteria in the marketing calendar application change (FilterManager event <code>CO_FM_EVT_FILTERS_CHANGED</code>). It collects the request payload for the OData request (entity(set), URL parameters, filters and sorters).</p> <p>The function <code>fnRequestPayloadCb</code> must return an object with the following properties:</p> <ul style="list-style-type: none"> • <code>Entity {string}</code> The entity (set) name on which the OData call is to be performed. This must be an entity(set) available in the OData model that was used when the <code>ODataBatchHandler</code> was instantiated. Otherwise the request will fail with an error. • <code>UrlParameters {object}</code> (optional) A mapping object with the following properties: <ul style="list-style-type: none"> ◦ <code>UrlParameters.Select {string[]}</code> (optional) An array of strings containing the property names requested from the OData entity for <code>\$select</code>. ◦ <code>UrlParameters.Expand {string[]}</code> (optional) An array of strings containing the navigation property names requested from the OData entity using <code>\$expand</code>. ◦ <code>UrlParameters.Skip {number}</code> (optional) The OData <code>\$skip</code> option value. ◦ <code>UrlParameters.Top {number}</code> (optional) The OData <code>\$top</code> option value ◦ <code>UrlParameters.Inlinecount {string}</code> (optional) The OData <code>\$inlinecount</code> option value. Can be "allpages" or "none". • <code>Filters {sap.ui.model.Filter[]}</code> (optional) An array of <code>{sap.ui.model.Filter}</code> objects. • <code>Sorters {sap.ui.model.Sorter[]}</code> (optional) An array of <code>{sap.ui.model.Sorter}</code> objects. <p>OData options not specifically mentioned are not supported. The data in the <code>UrlParameters</code> mapping object is automatically encoded. Erroneous values will cause OData errors as in a standard OData call.</p> <p>The function <code>fnSuccessCb</code> is called when the OData request specified with the <code>fnRequestPayloadCb</code> was processed without errors and passes the received data as a JSON object to the controller (identical to the success callback function in a standard <code>OData.read()</code> call). The combination of a view (controller) instance and the callback functions can only be registered once in the batch handler. If the caller tries to register a view twice, the existing registration is removed before the new one is added.</p> <p>Parameters:</p> <p><code>{sap.ui.controller} oController</code> Context object on which to call the functions.</p> <p><code>{function} fnRequestPayloadCb</code> The function to call to read the OData request payload. The function must return an object as described above.</p> <p><code>{function} fnSuccessCb</code> The function that is called when the OData request was processed successfully. It passes a JSON object with the response data to the function.</p>
<p>triggerODataRead()</p> <p>Trigger an OData read call for all registered views.</p>

Method Detail

unregisterView(oController, fnRequestPayloadCb, fnSuccessCb)

The unregister function for the registerView() function. Unregisters the view and its callback functions. Usually used in the onExit event of a view controller.

Parameters:

{sap.ui.controller} **oController**

Context object on which to call the functions.

{function} **fnRequestPayloadCb**

The function to call to read the OData request payload. The function must return an object as described above.

{function} **fnSuccessCb**

The function that is called when the OData request was processed successfully. It passes a JSON object with the response data to the function.

6.3.4 Customer-Defined Configuration

Use

You can use standard widgets and customer-defined widgets to create a customer-defined user interface that is available to all users as a default (and therefore replaces the standard UI interface). You can also specify the view that is used to display campaigns when the app is started (calendar view or list view).

To do so, you must use the following **controller extension**:

Table 60:

Controller	Hook
hpa.cei.mkt.cal.Component	extHookGetCustomerApp

You can define a customer-defined configuration for every device type. The “tablet”, “phone” and “desktop” device types are currently supported (names from sap.ui.Device.system.SYSTEMTYPE). If no customer-defined configuration is proposed for a device type by the hook, the standard SAP Configuration is used here instead. You must define tabs (with widgets) **and** parameters for each device type.

The customer-defined configuration can contain both SAP standard widgets and customer-defined widgets. The widgets can be distributed between any number of tabs that you define. You must define the attributes `Name`, `Icon` and `Widgets` for each tab. Customer-defined widgets must be defined using the hook `extHookGetCustomerWidgets` (See [Customer-Defined Widgets \[page 166\]](#)). The widgets available in the

standard SAP system are found in the `WidgetRepositoryManager` (File: `util.WidgetRepositoryManager.js`, function constructor). The following standard widgets are currently available:

Table 61:

Id	Name	Description
InitiativeSuccess	Campaign Success by Category	Shows the success of campaigns
IniCategoriesAndChannels	Top Categories + Top Channels	Shows the categories and channels of campaigns
SentimentMicroChart	Post Sentiment	A microchart showing sentiment
InterestMicroCharts	Top Interests (Posts) + Sentiment Score	A microchart showing interest
PlannedMarketingSpend	Planned Spend by Country	Shows the planned marketing spends by country
TopSpendTypesChart	Planned Spend	Shows the planned spend
AverageSentiment	Average Sentiment	Shows the average sentiment
PostsPerSentiment	Posts by Sentiment	Number of social media posts by sentiment

Display of campaigns in the bottom half of the app (calendar view or list view) is always available (meaning it does not have to be added using the hook).

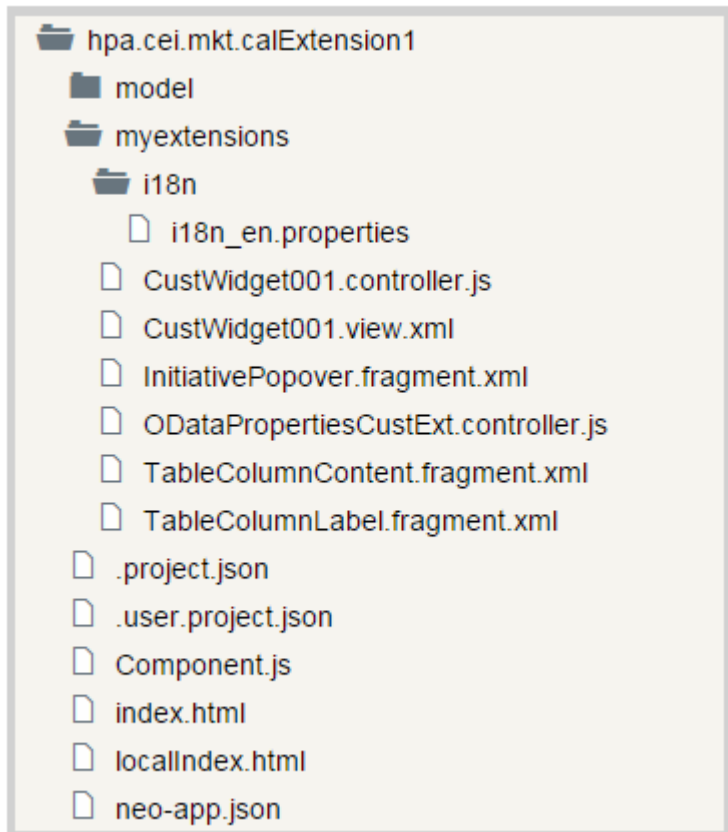
The hook must also be used to define the default parameters. The following parameters are currently supported:

Table 62:

Key	Description	Value
maxCampaignsShownInGantt	Maximum Number of Campaigns in Calendar View	<any number>
calendarViewStateOnInitialLoad	View of Campaigns (Calendar View or List View) each time the app is started	Gantt List

Example

The extension options available are explained using examples. The examples are based on the extension project **hpa.cei.mkt.calExtension** with the following structure:



Extensions to the File Component.js (in the Extension Project)

```
this.hpa.cei.mkt.cal.Component.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.Com
ponent", { metadata: { }, // metadata  extHookGetCustomerApp: function() {
return {      UserPersonalizationInjected: {      desktop: {
parameters: [{      key: "maxCampaignsShownInGantt",      value:
"200"      }, {      key: "calendarViewStateOnInitialLoad",
value: "List"      }],      tabs: [{      icon: "sap-icon://
favorite",      name: "My Favorites",      widgets: [{
id: "InitiativeSuccess"      }, {      id:
"CustWidget001"      }]}], {      icon: "sap-
icon://fallback",      name: "Spend & Category",      widgets:
[{      id: "TopSpendTypesChart"      }, {      id:
"IniCategoriesAndChannels"      }]}], //
desktop      phone: {      parameters: [{      key:
"maxCampaignsShownInGantt",      value: 10      },
{      key: "calendarViewStateOnInitialLoad",      value:
"Gantt"      }],      tabs: [{      icon: "sap-icon://
favorite",      name: "My Favorites",      widgets: [{
id: "InitiativeSuccess"      }, {      id:
"CustWidget001"      }]}], {      id:
phone      }      }; // return  } //extHookGetCustomerApp });
```

6.3.5 Deactivate Personalization

Use

You can deactivate the personalization if users should not have the option to configure the user interface to meet their specific requirements.

To do so, you must use the following **controller extension**:

Table 63:

Controller	Hook
hpa.cei.mkt.cal.Component	extHookSetPersonalizationUiEnabled

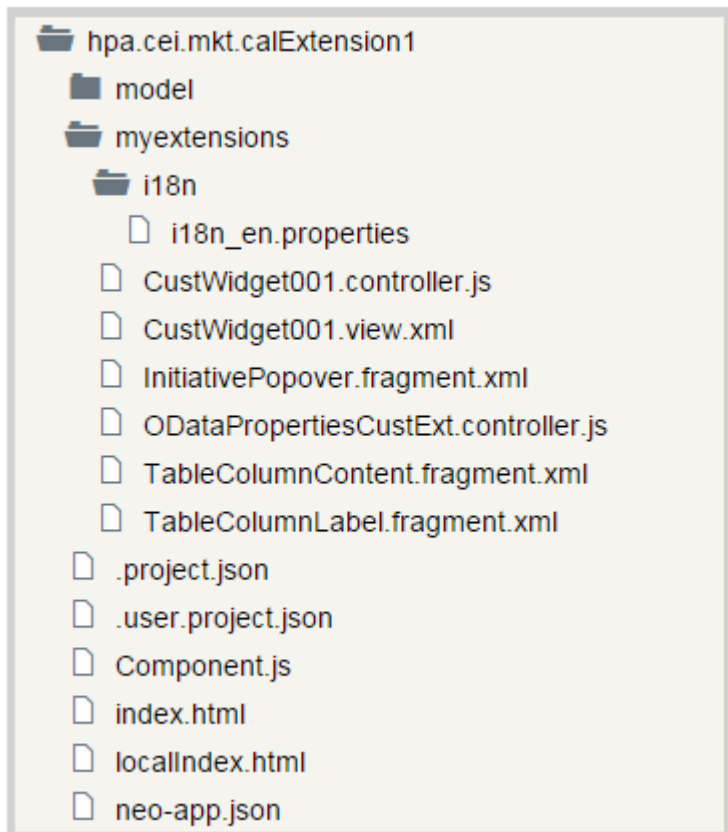
In this hook, you define whether the personalization is available or not. This function is active in the standard SAP system. You only need to implement the hook if you want to deactivate this function.

Note

If the personalization has been deactivated, any personalization data, which was created when the personalization was active, is still available in the system. You can delete this personalization data in the backend system using the transaction `/UI2/PERS_DEL` (Report `/UI2/PERS_EXPIRED_DELETE`). Use `hpa.cei.mkt.cal.pers.container` as the ID.

Example

The extension options are explained using examples. The examples are based on the extension project **hpa.cei.mkt.calExtension** with the following structure:



Extensions to the File Component.js (in the Extension Project)

```
this.hpa.cei.mkt.cal.Component.extend("hpa.cei.mkt.cal.hpa.cei.mkt.calExtension1.Component", {
  metadata: { }, // metadata
  extHookSetPersonalizationUiEnabled: function() {
    // to enable UI Personalization (default) --> return true;
    // to disable UI Personalization --> return false
    return false;
  } //extHookSetPersonalizationUiEnabled
});
```

6.3.6 More Information

- For general information about making extensions to SAP HANA views (Mass Copy) and OData Services, see [Extending the Data Source and Business Objects \[page 16\]](#).
- For more information about the creation and redefinition of OData Services, see <http://help.sap.com>. Choose **Technology** > **SAP Gateway**. Choose a release and then **Application Help** > **SAP NetWeaver Gateway Developer Guide** > **SAP NetWeaver Gateway Service Builder** > **Data Modeling Basics** > **Data Modeling Options** > **Redefining Services**.

-
- For information about the general procedure for extending SAP Fiori apps, (including UI extensions, deploying apps in the backend system, configuring the backend system), see SAP Fiori App Extensibility at https://websmp106.sap-ag.de/~sapidp/012002523100011849052014E/SAP_BP/BBLibrary/Documentation/MF8_NGW20_BB_ConfigGuide_EN_XX.doc
 - For a general description of the extensibility options and procedures of SAP Fiori apps, see ► *Extensibility Information for SAP Fiori* ► *Extensibility* in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.
 - For more information about extension includes, see ► *SAP Fiori for SAP Business Suite* ► *Extensibility Information for SAP Fiori* ► *Extensibility* ► *Checking the SAP-Enabled Extension Options* ► *Extension Includes* in the application help for SAP Fiori for SAP Business Suite on SAP Help Portal at <http://help.sap.com/fiori>.
 - For more information about the SAPUI5 Documentation, see <https://sapui5.hana.ondemand.com/sdk/#content/Overview.html>.

7 SAP Hybris Marketing Acquisition

7.1 Creating Customer-Specific Subscription Landing Pages

Caution

The specific programming interface (API) described in this section is no longer being improved. Instead, you should use the standard OData service `CUAN_IMPORT` to handle subscription creation and updates.

A subscription landing page is a Web page or a section on a Web page that provides the user with the option to give or reject general marketing permission for different communication media or to subscribe to your newsletters.

You can create a subscription landing page and position it on your Web site, for example. You can use this kind of landing page as a starting point for a simple or a double-opt in process including the Confirmation Messages application (for more information, see [Double Opt-In or Opt-Out Process](#) and [Defining Confirmation Messages](#) on the Help Portal under <http://help.sap.com/mkt> ► [Application Help](#) ► [SAP Hybris Marketing Worksets and Applications](#) ► [Campaigns](#) ► [Content Library, Template Library, Confirmation Messages](#) ►).

To create a subscription landing page, you are provided with API that you can use for communication with the [SAP Hybris Marketing](#) system. In the following, you are provided with a description of the interface and the corresponding process.

7.1.1 Internet Communication Framework

You need to activate the internet communication framework (ICF) node `/sap/public/cuan/cust_page` using the transaction SICF. This allows you to invoke the respective Web service in order to implement a subscription landing page on your Web site.

Caution

For security reasons, the ICF node is not activated by default and should only be activated after considering the security risks, such as denial of service attacks (DoS).

7.1.2 Background Information

With a subscription landing page on your Web site, you provide your Web users with a publicly available option to communicate with your [SAP hybris Marketing](#) system as you want them to submit their consent to receive emails

or newsletters, for example. For this reason, you are provided with Web services via the internet communication framework (ICF).

You need to configure the Web services to enable the Web user to send information to your Web server, which is saving the data to the database. On the Web server, a periodic background job sends the stored information in the correct form to the public service of the *SAP hybris Marketing* application server.

Another option is to allow the Web site to send the information entered by the Web users directly to the *SAP hybris Marketing* application server. Due to security reasons, this option is not recommended. If you decide for this option, you do it at your own risk.

7.1.3 Logic Flow

The ICF service `/cuan/cust_page` allows to send marketing newsletter and permission data to the *SAP hybris Marketing* system.

Using this service, you can implement your own Web pages offering newsletter subscription and permission granting options to a Web user. When the Web user submits the subscription to newsletters, for example, the corresponding request data or payload can be sent to the `/cuan/cust_page` ICF service. The service processes follow-on actions and saves corresponding interactions. Based on these interactions, the system stores the newsletter subscriptions or marketing permissions in the database.

SAP hybris Marketing will later use this new permission marketing data to send or not send emails to the Web user based on its input on the web page.

The `/cuan/cust_page` ICF service as well as all other ICF services described in this document are located under the path `/default_host/sap/public/cuan/` to not collide with other namespaces.

Request Handling

The ICF service accepts only HTTP POST requests in a specific JSON format.

Request Payload

The JSON format of the request payload is limited to 2000 characters and includes several attributes with general data about the specific user of the Web page as well as an array with the requested actions such as newsletter subscriptions.

The complete format is built up as follows:

- `Client` (String):
This field contains the ID of your productive client.
- `ContactId` (String):
This field contains external IDs of your Web user contacts such as the contacts' email addresses.

- **ContactIdOrigin (String):**
This field contains the origin of your contact IDs. The origin of an ID indicates the source of this ID such as [Email](#).
The allowed values depend on your Customizing entries in Customizing for [SAP hybris Marketing](#) under [Data Management](#) > [Interaction Contacts](#) > [Define Origins of Contact ID](#) . Usually, it is EMAIL.
- **LaunchURL (String):**
This field contains the starting point URL of an interaction such as the [Submit](#) pushbutton on a subscription landing page for interaction type NEWSLETTER_SUBSCR ([Subscribe to Newsletter](#)).
- **RemoteAddress (String):**
This field can contain the user IP address, which must be retrieved using your Web server.
- **Actions (Array):**
 - **Name (String):**
For the submission of general marketing permission information, the allowed values are the following:
 - OPTIN
 - OPTOUT
 For the subscription of a newsletter, the allowed values are the following:
 - SUBSCRIBE
 - UNSCUBSCRIBE
 - **Pre (Boolean):**
This boolean is set to false if you do not require a double opt-in process (usually for the submission of marketing permission information). It is set to true if you require a double opt-in process (usually for the subscription of newsletters).
Consider the use of the double **opt-out** process carefully: A Web user who wants to generally terminate receiving marketing emails or newsletters might be irritated when receiving another mail asking for approval of termination.
 - **OutboundCommunicationMedium (String):**
The allowed values depend on your Customizing entries in Customizing for [SAP hybris Marketing](#) under [Data Management](#) > [Interactions](#) > [Define Communication Media](#) . Usually, it is EMAIL.
 - **NewsletterId (String):**
This field is required for subscriptions only. It contains the key of the communication category for which the subscription should be saved (field `CommunicationCategoryKey`).

Example Code in JavaScript

In the following, you find a code example in JavaScript for your reference.

```

<html>
<head>
    <script src="http://code.jquery.com/jquery-2.1.4.min.js"></script>
    <script>
        function performSubscribe() {
            var data = {
                Client: "100",
                ContactId: $("#email_address").val(),
                ContactIdOrigin: "EMAIL",
                LaunchURL: window.top.location.href,
                Actions: [
                    {
                        // Example of a marketing permission opt-in
                        Name: "OPTIN", // or "OPTOUT"
                        Pre: false, // Set to false if no double opt-in process is needed
                        OutboundCommunicationMedium: "EMAIL" // or "PHONE" or ...
                    },
                    {
                        // Example of a newsletter pre-subscription
                        Name: "SUBSCRIBE", // or "UNSUBSCRIBE"
                        Pre: true, // Set to true if double opt-in process is needed
                        OutboundCommunicationMedium: "EMAIL",
                        NewsletterId: "ABCDEF" // = CommunicationCategoryKey
                    }
                ]
            };
        }
    </script>

```

```

jQuery.ajax({
    type: "POST",
    url: "https://{domain:port}/sap/public/cuan/cust_page",
    crossDomain : true,
    data: JSON.stringify(data),
    contentType: "application/json;charset=utf-8",
    dataType: "json",
    jsonp: false,
    success: function(mData) {
        document.write("Thank you for your submission!");
    },
    error: function($request) {
        alert($request.responseJSON.STATUS + ": " + $request.responseJSON.MESSAGE)
    }
});
}
</script>
</head>
<body>
    Please enter your email address and submit to get the latest information on our products.
    <br/><br/>
    <label for="email_address">Email Address:</label>
    <input id="email_address" type="text" maxlength="50">
    <br/><br/>
    <button onclick="performSubscribe()">Submit</button>
</body>
</html>

```

Request Response

The ICF service is responding with an own JSON format informing about success or error. Since the actual interactions are created asynchronously, the success response solely informs the client about correctness and completeness of the payload. If the payload is not correct or complete, the response includes information about which attribute is missing or invalid.

Background Job

For processing the provided data and saving the resulting marketing permissions and subscriptions, the report `CUAN_CONTENT_PAGE_EXECUTION` needs to be scheduled as a background job in your actual system client to be executed on a regular basis (for example, every minute).

Related Development Packages

The package `CUAN_CONTENT_PAGE_EXECUTION` contains the ICF service handler class for content pages, the corresponding background report, and the relevant dictionary objects for these classes.

Translation relevant objects associated to package `CUAN_CONTENT_PAGE_EXECUTION` such as message classes are included in package `CUAN_CONTENT_PAGE_EXEC_TRSL`.

7.2 Outbound Integration with SAP CRM

For more information about the outbound integration with SAP CRM, see SAP Service Marketplace <http://service.sap.com/mkt> then choose the required release and look for:

- Creating SAP CRM Business Transactions
- Checklist for the SAP CRM Integration Setup (see also the next section about automatic checks)

Note that you can use the following backend transactions for an automatic check of the required settings:

- In the SAP Hybris Marketing system:
 - SAP Hybris Marketing CRM Integration Check Report (`CUAN_CRM_CHECK`)
 - Condition Check (`CUAN_TC_CHECK`) in the folder Cuan CRM Integration Check.
- In the SAP CRM system: `CRM_CUAN_CHECK`: Consider [2202431](#) to enable the check.

7.3 Extend the Winner Determination for A/B Testing

You can extend the winner determination for the A/B testing in your campaign execution with own determinations and also remove, if required the existing ones.

1. Create a new class in a local implementation (**z**-class), using super class `CL_CUAN_MKT_EXEC_EXE_AB_EMAIL`.
2. Redefine method `IF_CUAN_MKT_EXEC_AB_TEST~GET_SUPPORTED_WINNER_DETERM.`
3. You can call the method of the super class to get the standard winner determinations.
4. Now you can decide if you want to:
 - add an own winner determination additionally to the standard determinations.
 - remove a standard winner determination without adding an own one.
 - add an own winner determination and remove the standard determinations.

Example

Use the key performance indicator (KPI) `RATE_OF_UNOPENED_MSSGS` of the SAP HANA view and remove the KPI `RATE_OF_OPENED_MSSGS` from the list of available winner determinations. Set order to `if_cuan_mkt_exec_ab_test=>sc_determination_type-lowest` to specify that the variant with the lowest rate of unopened messages should win.

Add as text of your own determination **Lowest Rate of Unopened Messages**. This text will be shown in the drop-down box of the *Winner Determination*.

Your implementation could look as follows:

Sample Code

```
INSERT VALUE #( id          = 'NO_OF_UNOPENED_MSSGS'
                 description = 'Number of unopened Messages'
                 order       = if_cuan_mkt_exec_ab_test=>sc_determination_type-lowest )
INTO TABLE rt_determ.
INSERT VALUE #( id          = mc_kpi_unique_click_rate
                 description = cl_cuan_mkt_exec_text_elements=>get_text( 5 )
                 order       = if_cuan_mkt_exec_ab_test=>sc_ab_winner_variant_det_order-highest )
INTO TABLE rt_determ.
```

In this example you do not need to redefine the method `IF_CUAN_MKT_EXEC_AB_TEST~DETERMINE_WINNER` because you use an existing KPI of the SAP HANA view and your logic is still to use the HIGHEST number for the winner determination.

Then in the Customizing for SAP Hybris Marketing under **► Campaigns ► Campaign ► Define Campaign Categories and Actions** you must replace the standard class by your **z**-class for action *A/B Testing* (`AB_TEST_EMAIL`) to get all the action parameters on the user interface.

Note

If you create an own action with an own ID, such as `Z_AB_TEST` in Customizing and assign here your own class, the user interface shows this new action without parameters.

7.4 Adding Own Actions for Campaign Automation

With the action framework, you can add actions with own business logic for your campaign automation.

The action framework provides classes and interfaces required for implementing new actions:

- Class `CL_CUAN_MKT_EXEC_EXECUTE_ACTN` or one of the subclasses such as `CL_CUAN_MKT_EXEC_EXECUTE_EMAIL`
- Interface `IF_CUAN_MKT_EXEC_ACTION_PARAM`
- Interface `IF_CUAN_MKT_EXEC_EXECUTE_ACTN`

Either the class or both interfaces are required for implementing the design-time as well as the runtime part of each action. Otherwise the new implementation will be rejected.

Each new action or adaption of existing actions requires a new implementation by using either the interfaces or by inheriting from an appropriate super class. The first two use cases demonstrate how to extend the functionality of existing actions whereas the last use case implements a new functionality.

The code snippets should give an impression how to implement the most important methods depending on different uses cases.

Necessary Steps to Implement a New Action

1. Implementation

○ Use Case 1

Overwrite the marketing permission check for standard action *Send Email* (Action ID `SEND_EMAIL`).

- Use class `CL_CUAN_MKT_EXEC_EXECUTE_EMAIL` as superclass.
- Adapt the method `CHECK_MARKETING_PERMISSION` (from class `CL_CUAN_MKT_EXEC_EXECUTE_ACTN`) according to your needs.

○ Use Case 2

Implement a new action parameter for standard action *Send Email* (Action ID `SEND_EMAIL`) and use it to enable or disable the marketing permission checks.

- Use class `CL_CUAN_MKT_EXEC_EXECUTE_EMAIL` as superclass.
- Adapt the method `SET_ACTION_PARAMETER` (from interface `IF_CUAN_MKT_EXEC_ACTION_PARAM`) according to your needs.
- Adapt the method `POST_PROCESS` (from interface `IF_CUAN_MKT_EXEC_EXECUTE_ACTN`) according to your needs.

○ Use Case 3

Send any personalized data to an external service consumer by using your own action (new action ID, starting with **z**).

- Use class `CL_CUAN_MKT_EXEC_EXECUTE_ACTN` as superclass.
- Adapt the methods `SET_ACTION_DETAILS` and `SET_ACTION_PARAMETER` (from interface `IF_CUAN_MKT_EXEC_ACTION_PARAM`) according to your needs.
- Adapt the methods `PRE_PROCESS` and `PROCESS` (from interface `IF_CUAN_MKT_EXEC_EXECUTE_ACTN`) according to your needs.

2. Customizing

After you have done the implementation you must activate it. Depending on the use case you must either set up a configuration for the new action or you change the implementation class for an existing action.

The action configuration is required for all three use cases, because each use case needs an action implementation class.

You find the configuration in the Customizing of SAP Hybris Marketing under ► [Campaigns](#) ► [Campaign](#) ► [Define Campaign Categories and Actions](#) ►.

There you do the following:

○ Use Case 1

For the *Action ID* `SEND_EMAIL` assign the class name of your own action implementation.

○ Use Case 2

For *Action ID* `SEND_EMAIL` assign the class name of your own action implementation.

○ Use Case 3

Define a new action ID, the ID must start with the character **z**, and assign the class name of your own action implementation.

Assign the action ID to an existing campaign category or create a new one.

i Note

In case you adapted a predefined action we strongly recommend to assign the adapted action to a new implementation class, because otherwise the action loses its parameters.

For more information, see:

[Implement Action for Design Time \[page 188\]](#)

To implement an action for the design time you need the following methods of the interface

IF_CUAN_MKT_EXEC_ACTION_PARAM: Method SET_ACTION_DETAILS for the action details and icon, and method SET_ACTION_PARAMETER for the action parameters.

[Implement Action for Run Time \[page 190\]](#)

To implement an action for the run time you need methods of the interface

IF_CUAN_MKT_EXEC_EXECUTE_ACTN. The action processing consists of the phases preprocess, process, and postprocess. For each phase a corresponding method exists to handle the required steps of the action in the corresponding phase.

[Reuse Common Logic by Inheriting \[page 195\]](#)

Action implementations can inherit from class CL_CUAN_MKT_EXEC_EXECUTE_ACTN and benefit from common reuse functionality for action implementations that is available there.

[Reuse Common Logic to Retrieve Personalization Data \[page 197\]](#)

Several classes and methods are delivered that retrieve personalization data and can be reused for new action implementations.

[Which Interactions To Create? \[page 197\]](#)

The basic concept in campaign automation is to have a single interaction for each outbound that reflects the outbound status.

[How to Enable a Restart in Case of Technical Failures? \[page 198\]](#)

In case an error occurs in your action logic, exit the PROCESS method with EV_ERROR = ABAP_TRUE.

7.4.1 Implement Action for Design Time

To implement an action for the design time you need the following methods of the interface

IF_CUAN_MKT_EXEC_ACTION_PARAM: Method SET_ACTION_DETAILS for the action details and icon, and method SET_ACTION_PARAMETER for the action parameters.

Method SET_ACTION_DETAILS

With this action you specify the action icon on the campaign automation user interface (UI).

You can either use a URL or an SAP UI5 icon URI. If you assign both, the internet icon gets the priority.

Example assigning an SAP UI5 icon:

```
es_action-icon_name = 'upload'.
```

Example using a hyperlink:

```
es_action-icon_url = 'https://wiki.wdf.sap.corp/wiki/download/attachments/1835069895/icon-upload.png'.
```


Method SET_ACTION_PARAMETER

In this method the list of action parameters are specified that shall be available on the UI:

- `action_parameter`: The ID of the parameter, must start with **Z**.

Note

It is not supported to use an SAP standard action parameter here.

- `action_parameter_name`: The label for the parameter used on the UI.
- `action_parameter_type`: Controls how the parameter is rendered on the UI. The following values are supported:
 - `Edm.String`: Parameter is rendered as a text field.
 - `Edm.Date`: Parameter is rendered as a date field.
 - `Edm.Time`: Parameter is rendered as a time field.
 - `Edm.Boolean`: Parameter is rendered as a checkbox.
- `action_parameter_values` (optional): Possible values for a parameter. If provided the parameter is rendered as dropdown list box.

Code Examples To...

... Define string, date, time, and Boolean parameter:

Sample Code

```
APPEND VALUE #( action_parameter = 'ZOC_EXPORT_DESCRIPTION'
                 action_parameter_name = 'Description'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-string ) TO et_action_parameter.
APPEND VALUE #( action_parameter = 'ZOC_EXPORT_DATE'
                 action_parameter_name = 'Date'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-date ) TO et_action_parameter.
APPEND VALUE #( action_parameter = 'ZOC_EXPORT_TIME'
                 action_parameter_name = 'Time'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-time ) TO et_action_parameter.
APPEND VALUE #( action_parameter = 'ZOC_CHECK'
                 action_parameter_name = 'Check'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-boolean ) TO et_action_parameter.
```

... Define a dropdown listbox with two entries:

Sample Code

```
APPEND VALUE #( action_parameter = 'ZOC_EXPORT_PRIORITY'
                 action_parameter_name = 'Priority'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-string
                 action_parameter_values = value #( ( action_parameter_value =
'PRIORITY_1'
                                                    action_parameter_value_name
= 'High' )
                                                    ( action_parameter_value =
'PRIORITY_2'
```

```

= 'Low' ) ) ) TO et_action_parameter.
action_parameter_value_name

```

... Add a new parameter to control the marketing permission check (use case 2):

Sample Code

```

CALL METHOD super->if_cuan_mkt_exec_action_param~set_action_parameter
IMPORTING
  et_action_parameter = et_action_parameter.
APPEND VALUE #( action_parameter      = |Z_IGNORE_MKTG_PRMSN|
                 action_parameter_name = 'Ignore Marketing Permission Check'
                 action_parameter_type =
if_cuan_mkt_orch_constants=>action_param_type-boolean ) TO et_action_parameter.

```

7.4.2 Implement Action for Run Time

To implement an action for the run time you need methods of the interface `IF_CUAN_MKT_EXEC_EXECUTE_ACTN`. The action processing consists of the phases preprocess, process, and postprocess. For each phase a corresponding method exists to handle the required steps of the action in the corresponding phase.

Method `PRE_PROCESS`

This method is called once before package processing and parallelization. It is responsible for providing metadata that is needed throughout the execution process. By this The campaign execution avoids repeated database access and common business logic in each package for metadata.

The method is intended for the action implementation to retrieve process relevant parameter values before the packaging and parallelization of the core business logic starts. The following exporting parameters are available:

- Flag to indicate whether the action uses parallelization at all.
- Action specific package size that overwrites default values.
- Marketing content attributes for personalization during processing.
- Maximum number of parallel tasks if more free tasks would be available in the system or server group.
- Process relevant parameter values
- Messages
- Indicate error during preprocess to stop further execution process.

Code Example

Example setting a parameter and using an export definition for personalization (use case 3):

Sample Code

```

CONSTANTS: lv_export_definition_id TYPE hpa_v_export_definition_id VALUE
'19818'. " Use a predefined export definition
DATA: lt_ed key TYPE /iwbp/t_mgw_tech_pairs.
ev_no_parallelization = abap_false.

```

```

ev_package_size      = 10.
ev_max_parallel_tasks = 5.
* Set parameters (e.g. retrieve RFC destination and pass for further processing)
APPEND VALUE #( param_name = 'RFC_DESTINATION'
                 param_value = 'Z_GUIDELINE_EXAMPLE' ) TO et_param.
* Get the export definition data
me->ms_export_definition = get_export_definition(
  EXPORTING
    iv_definition_id = lv_export_definition_id
  IMPORTING
    ev_error          = ev_error ).
IF ev_error EQ abap_true.
* Set parameter ET_MESSAGE of course
RETURN.
ENDIF.
APPEND VALUE #( name = if_hpa_export_def_common_c=>co_prop_definition_id
                 value = me->ms_export_definition-definition_id ) TO lt_ed_key.
* Get export definition attributes
TRY.
  NEW cl_hpa_export_def_data_prov( )->defattributes_get_entityset(
    EXPORTING
      iv_entity_type_name =
if_hpa_export_def_common_c=>co_et_definition_attribute
      it_keys              = lt_ed_key
    IMPORTING
      et_entityset         = DATA(lt_export_definition_attribute) ).
* Set table ET_CONTENT_ATTR
  LOOP AT lt_export_definition_attribute REFERENCE INTO
DATA(lr_export_definition_attribute).
    APPEND VALUE #( ds_attribute_name      = |DA-
{ lr_export_definition_attribute->at_ds_alias }-{ lr_export_definition_attribute-
>at_name }|
                    data_source_alias      = lr_export_definition_attribute-
>at_ds_alias
                    attrib_name            = lr_export_definition_attribute-
>at_name
                    attrib_as_description   = lr_export_definition_attribute-
>at_as_descr
                    attrib_col_sequence    = lr_export_definition_attribute-
>at_pos
                    attrib_keep_duplicates = lr_export_definition_attribute-
>at_keep_duplicates ) TO et_content_attr.
  ENDLLOOP.
  CATCH /iwbe/cx_mgw_tech_exception .
* Set parameter ET_MESSAGE of course
  ev_error = abap_true.
  CATCH /iwbe/cx_mgw_busi_exception .
* Set parameter ET_MESSAGE of course
  ev_error = abap_true.
ENDTRY.

```

Method PROCESS

This method realizes package processing and parallelization. It represents the method that realizes the core action business logic according to the settings and parameters.

The following parameters are imported:

- Execution run
- Package of target group members

- Marketing content attributes determined by PRE_PROCESS
- Process-relevant parameters determined by PRE_PROCESS

Exported are:

- Messages
- Error Flag

Code Example

Example executing the personalization by using the content attributes (use case 3):

Sample Code

```
DATA: lr_key_table          TYPE REF TO data,
      lt_failed_key         TYPE /bobf/t_frw_key,
      lt_action_parameter TYPE cuan_t_marketing_orc_act_par.
* Retrieve additional campaign data
read_dep_data( EXPORTING iv_exec_run_key = iv_exec_run_key
                it_tg_member = it_tg_member
                IMPORTING ev_error = ev_error ).
IF ev_error IS NOT INITIAL.
  et_message = me->get_messages( ).
  RETURN.
ENDIF.
mv_segmentation_object = ms_mktorc_root-segmentation_object.
* Retrieve action parameters
DATA(lo_srv_mktorc) = /bobf/
cl_tra_serv_mgr_factory=>get_service_manager( iv_bo_key =
if_cuan_marketing_orch_c=>sc_bo_key ).
lo_srv_mktorc->retrieve_by_association(
  EXPORTING
    iv_node_key = if_cuan_marketing_orch_c=>sc_node-execution_run
    iv_association = if_cuan_marketing_orch_c=>sc_association-execution_run-
to_parent
    it_key = VALUE #( ( key = iv_exec_run_key ) )
    iv_fill_data = abap_false
  IMPORTING
    et_failed_key = lt_failed_key
    et_target_key = DATA(lt_action_key) ).
IF lines( lt_failed_key ) > 0.
* Set parameter ET_MESSAGE of course
ev_error = abap_true.
RETURN.
ENDIF.
lo_srv_mktorc->retrieve_by_association(
  EXPORTING
    iv_node_key = if_cuan_marketing_orch_c=>sc_node-action
    iv_association = if_cuan_marketing_orch_c=>sc_association-action-
action_parameter
    it_key = lt_action_key
    iv_fill_data = abap_true
  IMPORTING
    et_failed_key = lt_failed_key
    et_data = lt_action_parameter ).
IF lines( lt_failed_key ) > 0.
* Set parameter ET_MESSAGE of course
ev_error = abap_true.
RETURN.
ENDIF.
* Table LT_ACTION_PARAMETER contains all parameters created in method
SET_ACTION_PARAMETER
* Convert target group member to segmentation key structure
convert_tg_members( EXPORTING it_tg_member = it_tg_member
                    IMPORTING er_key_table = lr_key_table
```

```

et_ic_key          = DATA(lt_ic_key)
et_ia_key          = DATA(lt_ia_key)
et_root_ia_key     = DATA(lt_root_ia_key)
et_object_type_key = DATA(lt_segobj_key)
et_object_type_ds_key = DATA(lt_segobj_ds_key)
ev_error           = ev_error
ev_ic_comp_name     = DATA(lv_ic_comp_name)
ev_ia_comp_name     = DATA(lv_ia_comp_name) ).

IF ev_error IS NOT INITIAL.
    et_message = me->get_messages( ).
    RETURN.
ENDIF.
*   Get personalization data
    cl_cuan_mkt_exec_helper=>get_dynamic_content( EXPORTING iv_tg_guid
= ms_tg_root-key
                                                    ir_key_table
= lr_key_table
                                                    iv_seg_object_type
= mv_segmentation_object
                                                    it_segobj_key
= lt_segobj_key
                                                    it_segobj_ds_key
= lt_segobj_ds_key
                                                    it_content_attr
= it_content_attr
                                                    iv_ic_comp_name
= lv_ic_comp_name
                                                    iv_ia_comp_name
= lv_ia_comp_name
IMPORTING et_exec_member_status
          et_pers_content
          et_messages
          ev_error
= DATA(lt_exec_member_status)
= DATA(lt_pers_content)
= et_message
= ev_error ).
IF ev_error IS NOT INITIAL.
    RETURN.
ENDIF.
*   Table LT_PERS_CONTENT consists of all target group members and their
personalization attributes (DYNAMIC_CONTENT)
*   Call your service consumer here. Set EV_ERROR and ET_MESSAGE in case of errors

```

Method POST_PROCESS

This method is called once after package processing and parallelization. It finalizes the execution such as write a file to a share after the core action logic in the PROCESS method has been executed.

Code Example

Example for using the new parameter to write a message into the application log (use case 2):

Sample Code

```

DATA: lt_action_parameter TYPE cuan_t_marketing_orc_act_par,
      lv_message           TYPE string.
CLEAR: et_message, ev_error.
*   // Call super class
CALL METHOD super->if_cuan_mkt_exec_execute_actn~post_process

```

```

EXPORTING
  iv_exec_run_key = iv_exec_run_key
  it_param        = it_param
IMPORTING
  et_message      = et_message
  ev_error        = ev_error.
* // Own implementation may overrule the standard check
IF mv_exec_run_key IS NOT INITIAL.
* // Retrieve action parameters by using the execution run key
DATA(lo_srv_mktorc) = /bobf/
cl_tra_serv_mgr_factory=>get_service_manager( iv_bo_key =
if_cuan_marketing_orch_c=>sc_bo_key ).
lo_srv_mktorc->retrieve_by_association(
  EXPORTING
    iv_node_key      = if_cuan_marketing_orch_c=>sc_node-execution_run
    iv_association    = if_cuan_marketing_orch_c=>sc_association-execution_run-
to_parent
    it_key            = VALUE #( ( key = mv_exec_run_key ) )
  IMPORTING
    et_target_key     = DATA(lt_action_key) ).
lo_srv_mktorc->retrieve_by_association(
  EXPORTING
    iv_node_key      = if_cuan_marketing_orch_c=>sc_node-action
    iv_association    = if_cuan_marketing_orch_c=>sc_association-action-
action_parameter
    it_key            = lt_action_key
    iv_fill_data      = abap_true
  IMPORTING
    et_data           = lt_action_parameter ).
READ TABLE lt_action_parameter REFERENCE INTO DATA(lr_action_parameter)
WITH KEY parameter_id = |Z_IGNORE_MKTG_PRSSN|.
IF sy-subrc EQ 0.
  IF lr_action_parameter->parameter_value EQ abap_true. "/// Check value of
parameter Z_IGNORE_MKTG_PRSSN
    MESSAGE i002(zcuan_mkt_exec_frwl) INTO lv_message. "/// Marketing
permission check is not active
  ELSE.
    MESSAGE i001(zcuan_mkt_exec_frwl) INTO lv_message. "/// Marketing
permission check is active
  ENDIF.
ELSE. "/// Parameter Z_IGNORE_MKTG_PRSSN not found
  MESSAGE i001(zcuan_mkt_exec_frwl) INTO lv_message. "/// Marketing
permission check is active
ENDIF.
cl_cuan_mkt_exec_messages=>add_message( CHANGING ct_messages = et_message ).
ENDIF.

```

Method EXCLUDE_MEMBERS_FOR_RESTART

This method is called in case of an execution run is restarted. Here the target group members that have been successfully processed in a previous run can be identified and excluded from a repeated execution.

7.4.3 Reuse Common Logic by Inheriting

Action implementations can inherit from class `CL_CUAN_MKT_EXEC_EXECUTE_ACTN` and benefit from common reuse functionality for action implementations that is available there.

The class `CL_CUAN_MKT_EXEC_EXECUTE_ACTN` provides protected member attributes that can hold values of dependent objects such as orchestration, target groups, and marketing content.

The basic concept of an action implementation is to realize the action-specific steps by a combination of own as well as common reuse methods. The data flow is based on an internal table of type `CUAN_T_MKT_EXEC_MEMBER_STATUS`. The underlying proposed methodology is to take this internal table through the steps of an action and enrich it with data needed for interaction creation, for example, failures in checks, outbound sending, and success outbound interactions.

The following reuse methods are available:

- `MKT_PERM_CHECK_IS_ACTIVE`
Determines whether marketing permission checks shall be done during the execution based in the campaign customizing settings.
Example for using the new parameter to control the marketing permission check (use case 2):

Sample Code

```
DATA: lt_action_parameter TYPE cuan_t_marketing_orc_act_par.
* // Call super method
CALL METHOD super->mkt_perm_check_is_active
  RECEIVING
    rv_active = rv_active.
* // Own implementation may overrule the standard check
IF mv_exec_run_key IS NOT INITIAL.
* // Retrieve action parameters by using the execution run key
  DATA(lo_srv_mktorc) = /bobf/
  cl_tra_serv_mgr_factory=>get_service_manager( iv_bo_key =
    if_cuan_marketing_orch_c=>sc_bo_key ).
  lo_srv_mktorc->retrieve_by_association(
    EXPORTING
      iv_node_key      = if_cuan_marketing_orch_c=>sc_node-execution_run
      iv_association   = if_cuan_marketing_orch_c=>sc_association-
        execution_run-to-parent
      it_key           = VALUE #( ( key = mv_exec_run_key ) )
    IMPORTING
      et_target_key    = DATA(lt_action_key) ).
  lo_srv_mktorc->retrieve_by_association(
    EXPORTING
      iv_node_key      = if_cuan_marketing_orch_c=>sc_node-action
      iv_association   = if_cuan_marketing_orch_c=>sc_association-action-
        action_parameter
      it_key           = lt_action_key
      iv_fill_data     = abap_true
    IMPORTING
      et_data          = lt_action_parameter ).
  READ TABLE lt_action_parameter REFERENCE INTO DATA(lr_action_parameter)
  WITH KEY parameter_id = |Z_IGNORE_MKTG_PRMSN|.
  IF sy-subrc EQ 0 AND lr_action_parameter->parameter_value EQ abap_true.
    "/// Own parameter found, check value
    rv_active = abap_false.
  ENDIF.
ENDIF.
```

- `IS_NEWSLETTER_CAMPAIGN`
Determines whether the campaign under consideration is a subscription-based campaign.

- **CHECK_MARKETING_PERMISSION**

Checks whether communication data for the contact is provided and marketing permission is given for the communication medium such as email and text message. In case of missing data or missing permission appropriate data for the interaction is added to the internal member status table.

Redefine only method `CHECK_MARKETING_PERMISSION` for use case 1:

Sample Code

```
* // Call superclass if required
CALL METHOD super->check_marketing_permission
EXPORTING
    iv_comm_medium    = iv_comm_medium
    iv_reason_failed  = iv_reason_failed
CHANGING
    ct_member_status = ct_member_status.
* // Implement your own checks
LOOP AT ct_member_status REFERENCE INTO DATA(lr_member_status).
    ...
    lr_member_status->ia_type = ...
    lr_member_status->ia_reason = ...
ENDLOOP.
```

- **GET_CAMPAIGN_CONTENT / GET_EXPORT_DEFINITION / GET_SEGMENTATION_PROFILE**

Reads the data of the corresponding business objects.

- **CONVERT_TG_MEMBER_TO_SEGKEY**

Converts the target group member keys that are handed over to the `PROCESS` method in the case of initial triggers with target groups into the keys as needed by the segmentation object, for example, contact key, interaction key, and product key. These keys can then be used to retrieve the dynamic content using the corresponding segmentation APIs.

- **CONVERT_INTERNAL_TG_TO_SEGKEY**

In case of trigger-based campaign or follow-up action, the action needs to process internal target groups. This method converts the keys of such internal target groups into keys as determined by the segmentation object. These keys can then be used to retrieve the dynamic content using the corresponding segmentation APIs.

- **CONVERT_TG_MEMBERS**

Determines for a list of target group members the corresponding keys:

- Segmentation keys as defined by segmentation object for retrieval of dynamic content
- Interaction contact keys
- Interaction keys
- Root interaction keys
- Meta-data on segmentation keys from segmentation object

This method internally uses the two previous methods.

The determined information can be used to retrieve the business data needed in action implementations.

- **INITIALIZE_MEMBER_STATUS**

As described above, we propose to base the data flow in an action implementation on an internal table of type `CUAN_T_MKT_EXEC_MEMBER_STATUS` to represent the status and interactions that are written for the list of target group members. This method initializes such an internal table, for example, by creating an entry for each target group member.

In case personalization attributes such as first and last name, are provided, an internal table for this personalization content is initialized. The table contains the personalization attributes for each target group member. The follow-up processing is done by the action implementation.

- **POST_INTERACTIONS**

Creates interactions according to the data in the internal member status table that is imported.

A template interaction can be provided, only those fields left empty in the template are enriched for the actual interaction creation.

Interactions are created without an update of the interaction contact besides last interaction timestamp.

- `READ_DEP_DATA`

Reads dependent data into member variables that can be accessed by the action implementation:

- BOPF campaign root
- BOPF target group root
- BOPF TG members keys
- BOPF execution run key

- `*_MESSAGES`

Convenience methods for message handling during the action processing.

7.4.4 Reuse Common Logic to Retrieve Personalization Data

Several classes and methods are delivered that retrieve personalization data and can be reused for new action implementations.

- Interaction contact root data: Class `CL_CUAN_MKT_EXEC_PERS_DATA_IC`
- Interaction contact team member data: Class `CL_CUAN_MKT_EXEC_PERS_DATA_TM`
- Dynamic content data based on segmentation object: Method `CL_CUAN_MKT_EXEC_HELPER > GET_DYNAMIC_CONTENT`

For the classes `CL_CUAN_MKT_EXEC_PERS_DATA_IC` and `CL_CUAN_MKT_EXEC_PERS_DATA_TM` the factory class `CL_CUAN_MKT_EXEC_FACTORY` can be used to create class instances as required by the list of personalization attributes. The syntax of the attribute ID is as defined by the campaign content and follows the pattern `<BO>-<Datasource>-<Attribute>`.

7.4.5 Which Interactions To Create?

The basic concept in campaign automation is to have a single interaction for each outbound that reflects the outbound status.

This ensures that a consistent reporting is possible. Otherwise it would be difficult to identify the interaction that represents the actual outbound status.

We recommend to use the following generic interaction types:

- `OUTBOUND_FAILED` for technical failures such as contact information is missing like email and no outbound is possible. We recommend to use appropriate reason codes.
- `OUTBOUND_CHK_FAILED` for business check failures such as failed permission checks or contact limit checks. We recommend to use appropriate reason codes.

These generic interactions are written if you use the method `POST_INTERACTIONS`.

In addition, you shall create scenario-specific additional outbound interactions in case of successful outbound actions.

7.4.6 How to Enable a Restart in Case of Technical Failures?

In case an error occurs in your action logic, exit the `PROCESS` method with `EV_ERROR = ABAP_TRUE`.

With this exit the execution status is set to erroneous and a restart on the UI is possible.

The standard restart implementation is based on interactions. For target group members with existing interactions no reprocessing is done. Only those are reprocessed for which no interactions have been written in the aborted execution run.

This behavior can be overwritten using method `EXCLUDE_MEMBERS_FOR_RESTART`.

7.5 Custom Fields in Campaign

You can use this option when you have a requirement to have additional fields in Campaign, along with SAP standard fields

Prerequisites

Based on your requirements, you have created custom fields for the business context [Marketing: Campaign](#). Ensure that you enable it for usage in the [UIs and Reports](#) tab for the data source `CUAN_CAMPAIGN_MDL`. Once you publish these fields, they are available in the flow-based [Campaigns](#) app.

Procedure

1. Navigate to the flow-based [Campaigns](#) app.
2. Create a new campaign or edit an existing one.
3. In the [Campaign Designer](#) screen, select the `CAMPAIGN` node.
4. In the right pane, navigate to the [Custom Details](#) section.
The custom fields that you defined in the [Custom Fields and Logic](#) app, appear under this section.
5. Enter the values for the custom fields.
6. Release the campaign.

Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: <http://help.sap.com/disclaimer>).



**go.sap.com/registration/
contact.html**

© 2017 SAP SE or an SAP affiliate company. All rights reserved.
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.
Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.
These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.
SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.
Please see <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.