Application Development with CDS + BOPF + ODATA + Fiori Elements (SAPUI5)

Purpose

This blog will focus on End-to-End development of an application using

1. CDS (Core Data Services)
    - Interface and Consumption View
    - BOPF Generation
2. BOPF (Business Object Processing Framework)
    - Business Processing Logic (CRUD Operations)
3. OData (Open Data Protocol)
    - Exposure of Data
4. SAPUI5 / Fiori Elements
    - Front End Development

Requirement

Build a Phone Book Application which helps to manage contacts. The initial version will support only basic features such as Create, Edit, Delete and Copy contacts. It will list all contacts in initial view with contact information such as First & Last Name, E-Mail ID. When the user clicks on any

record then detail view will display all basic information along with different telephone numbers for the selected contact.

Step By Step Guide

1. Design / Prototype
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

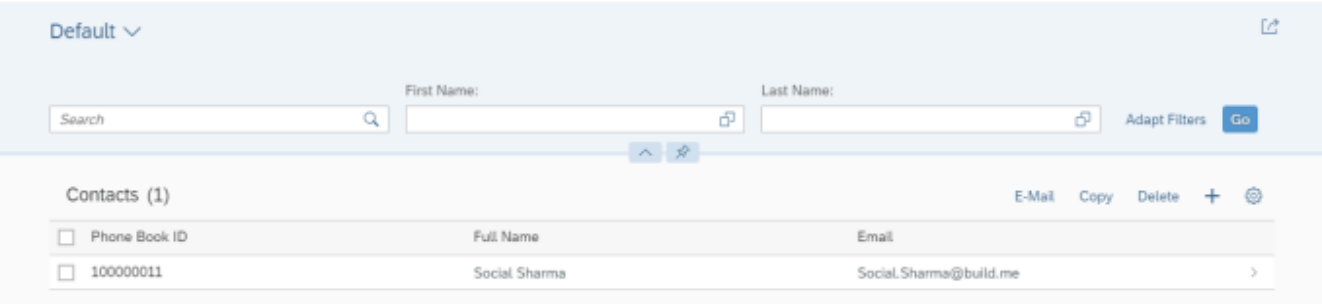
Design

---

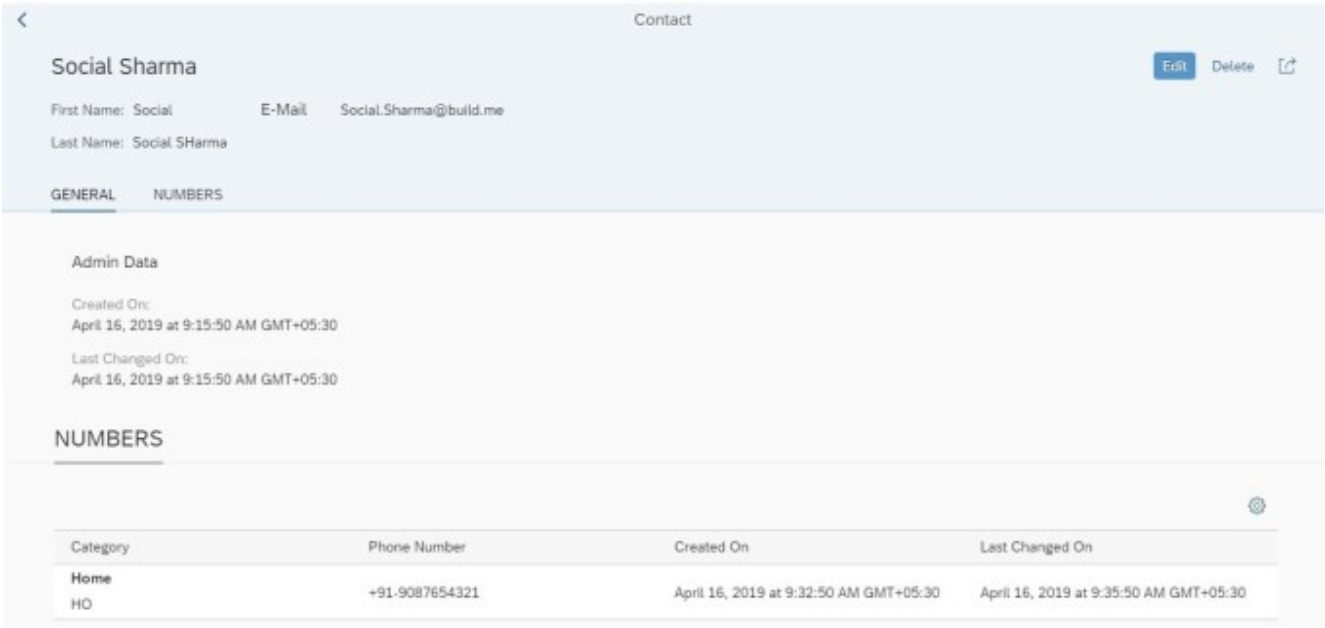
1. **Design / Prototype**
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

The design was created using SAP Build, you can preview the app using the below link.
PREVIEW @Phone Book (Prototype)

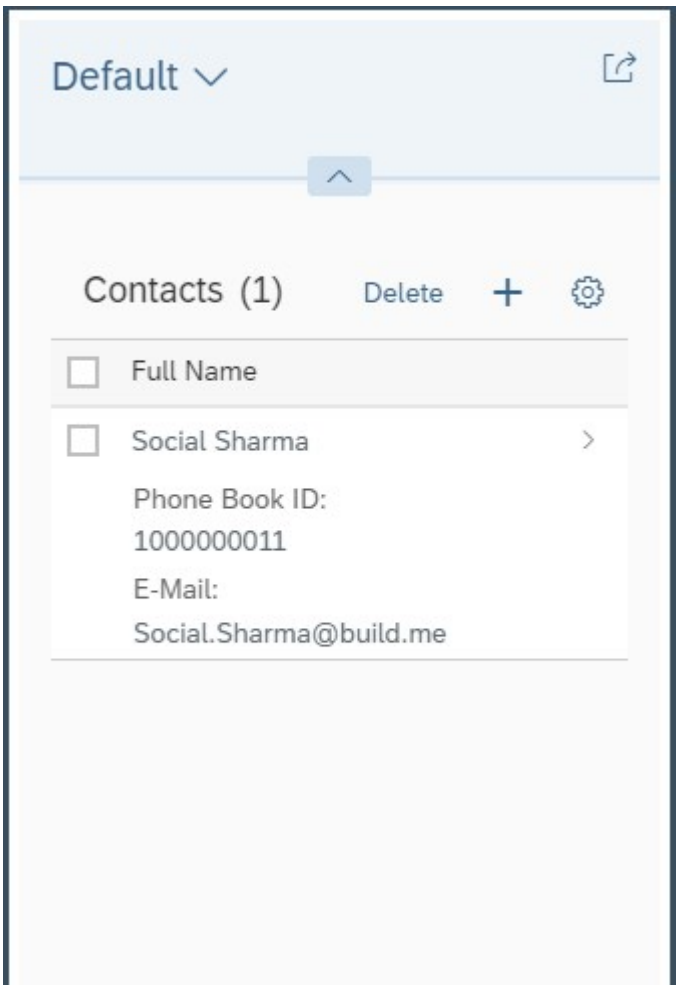
Desktop – List Report



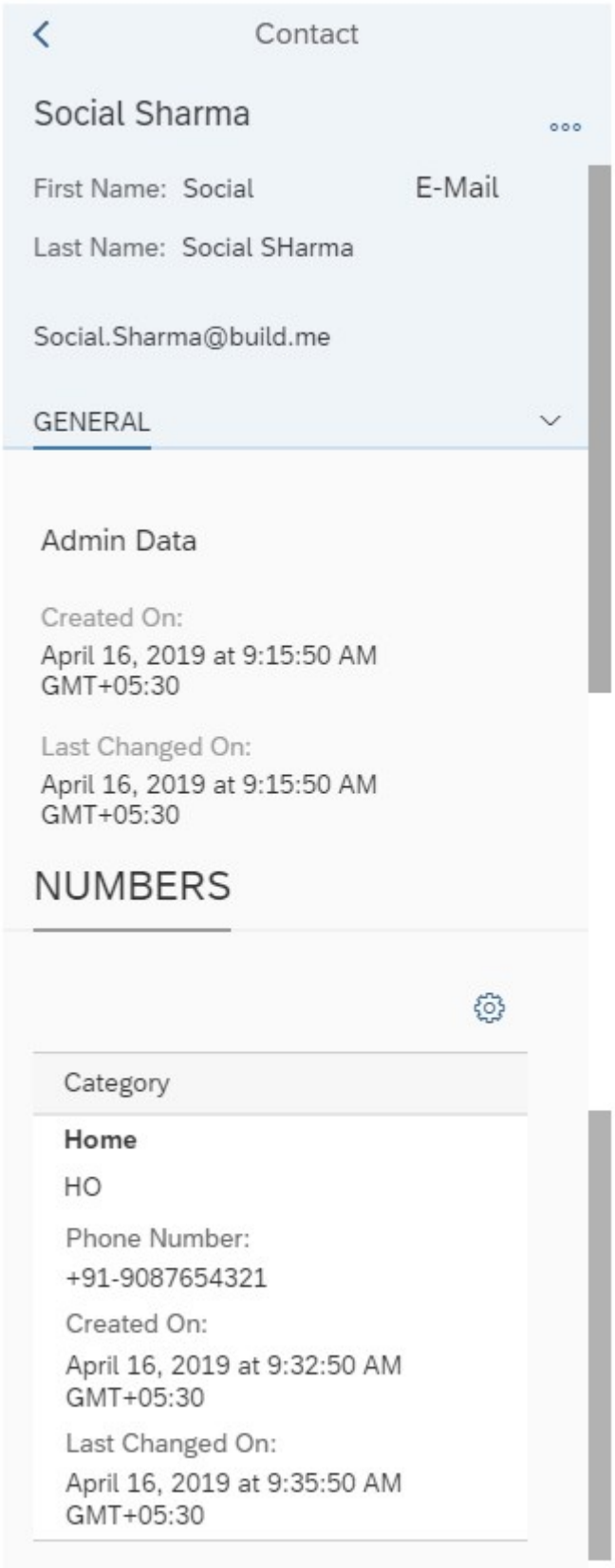
Desktop – Object Page



Mobile – List Report



Mobile – Object Page


Contact
Social Sharma
First Name: Social
E-Mail
Last Name: Social SHarma
Social.Sharma@build.me
GENERAL
Admin Data
Created On:
April 16, 2019 at 9:15:50 AM
GMT+05:30
Last Changed On:
April 16, 2019 at 9:15:50 AM
GMT+05:30
NUMBERS
Category
Home
HO
Phone Number:
+91-9087654321
Created On:
April 16, 2019 at 9:32:50 AM
GMT+05:30
Last Changed On:
April 16, 2019 at 9:35:50 AM
GMT+05:30

[<< Top](#)

Create Dictionary Objects

1. Design / Prototype
2. **Create Dictionary Objects**
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

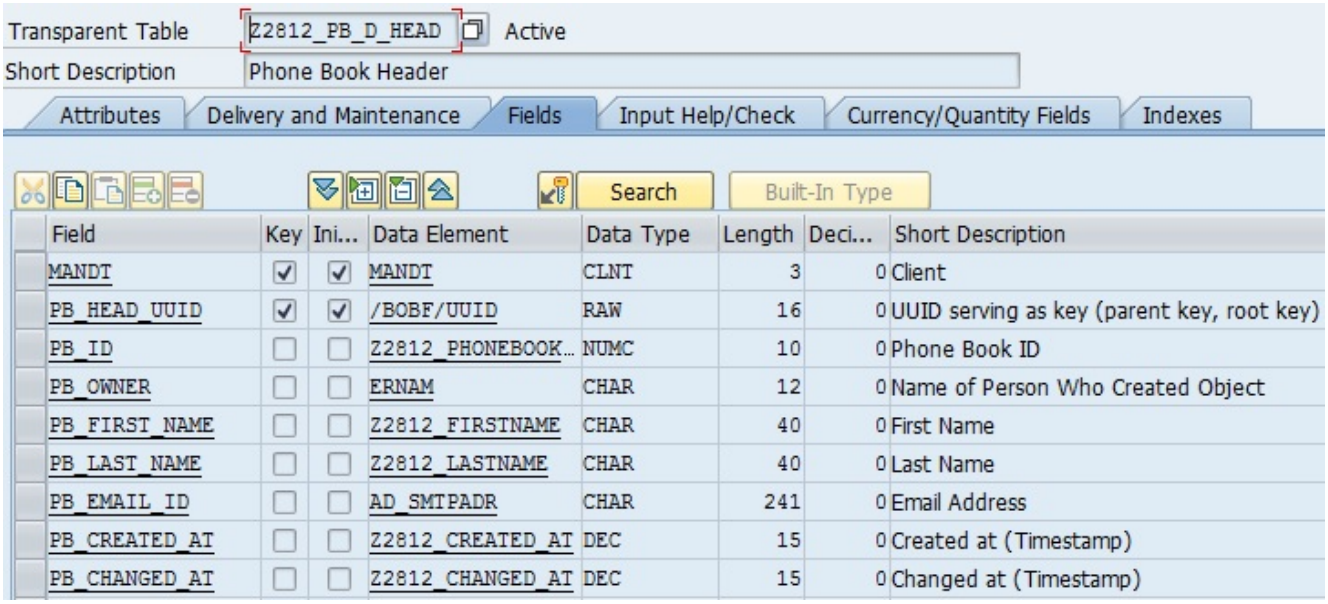
As per the requirement, we will need three table. One each for Header & Item data for contacts. Additionally, I created a master table for contacts category, this is required to supply value help for UI.

You can create dictionary table as per your requirement. Just make sure to use UUID as key.

Note:

Tables for storing Drafts will be auto generated with business objects, so there is no need to create it manually.
Created By and Changed By is not added as field, because Phone Book application is user based and it will have only one authorized owner, who can manage it.

Header Table for Phone Book



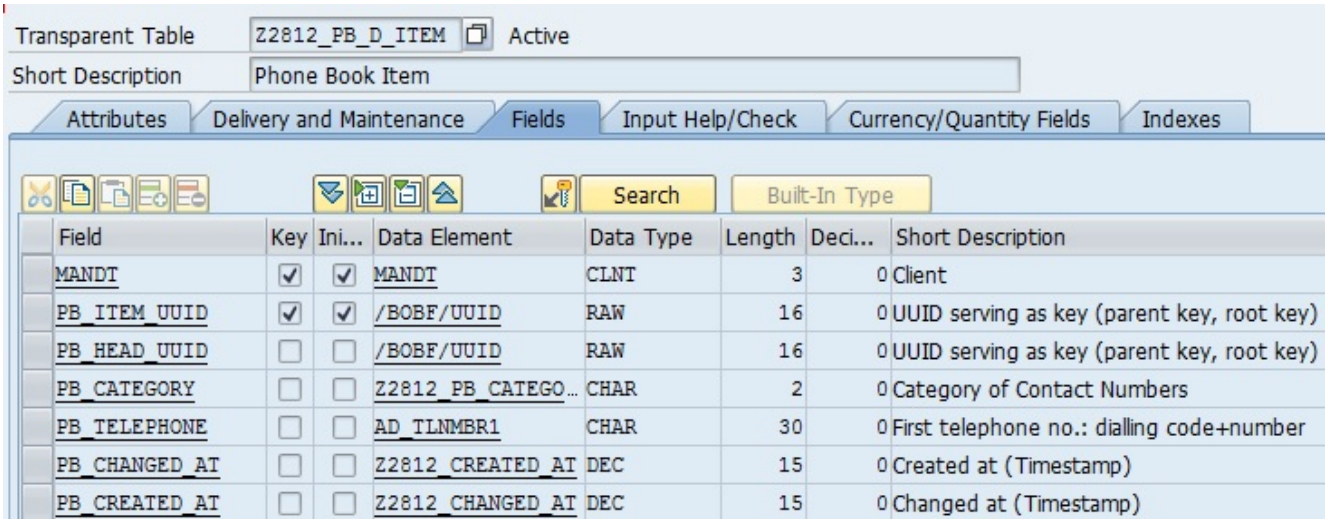
Transparent Table: Z2812_PB_D_HEAD — Active
Short Description: Phone Book Header

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexes

Search | Built-In Type

Field	Key	Ini...	Data Element	Data Type	Length	Deci...	Short Description
MANDI	☑	☑	MANDI	CLNT	3	0	Client
PB_HEAD_UUID	☑	☑	/BOBF/UUID	RAW	16	0	UUID serving as key (parent key, root key)
PB_ID	☐	☐	Z2812_PHONEBOOK...	NUMC	10	0	Phone Book ID
PB_OWNER	☐	☐	ERNAM	CHAR	12	0	Name of Person Who Created Object
PB_FIRST_NAME	☐	☐	Z2812_FIRSTNAME	CHAR	40	0	First Name
PB_LAST_NAME	☐	☐	Z2812_LASTNAME	CHAR	40	0	Last Name
PB_EMAIL_ID	☐	☐	AD_SMTPADR	CHAR	241	0	Email Address
PB_CREATED_AT	☐	☐	Z2812_CREATED_AT	DEC	15	0	Created at (Timestamp)
PB_CHANGED_AT	☐	☐	Z2812_CHANGED_AT	DEC	15	0	Changed at (Timestamp)

Item Table for Phone Book



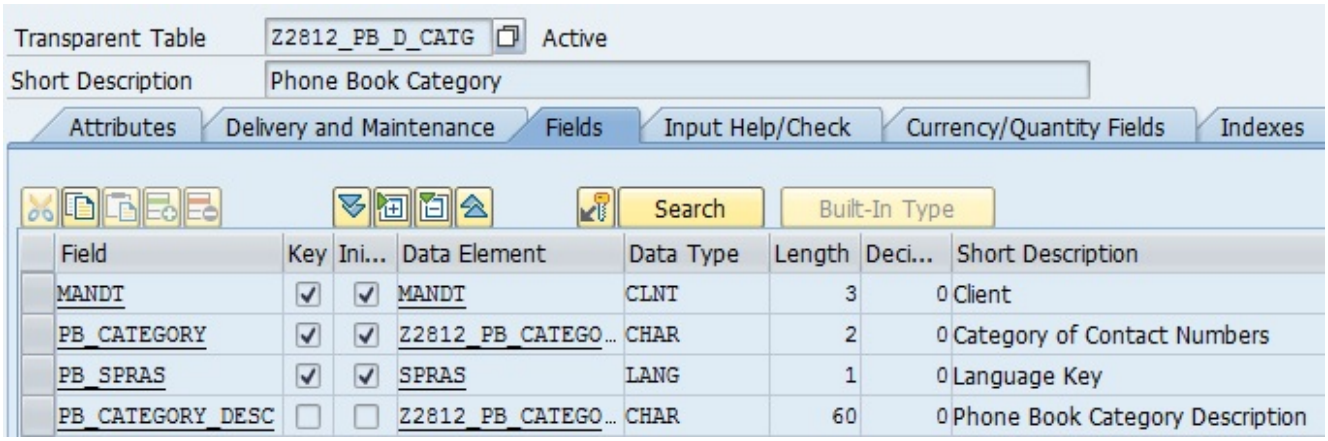
Transparent Table: Z2812_PB_D_ITEM — Active
Short Description: Phone Book Item

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexes

Search | Built-In Type

Field	Key	Ini...	Data Element	Data Type	Length	Deci...	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
PB_ITEM_UUID	☑	☑	/BOBF/UUID	RAW	16	0	UUID serving as key (parent key, root key)
PB_HEAD_UUID	☐	☐	/BOBF/UUID	RAW	16	0	UUID serving as key (parent key, root key)
PB_CATEGORY	☐	☐	Z2812_PB_CATEGO…	CHAR	2	0	Category of Contact Numbers
PB_TELEPHONE	☐	☐	AD_TLNMBR1	CHAR	30	0	First telephone no.: dialling code+number
PB_CHANGED_AT	☐	☐	Z2812_CREATED_AT	DEC	15	0	Created at (Timestamp)
PB_CREATED_AT	☐	☐	Z2812_CHANGED_AT	DEC	15	0	Changed at (Timestamp)

Contact Category Master Table



Transparent Table: Z2812_PB_D_CATG — Active
Short Description: Phone Book Category

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexes

Search | Built-In Type

Field	Key	Ini...	Data Element	Data Type	Length	Deci...	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	Client
PB_CATEGORY	☑	☑	Z2812_PB_CATEGO…	CHAR	2	0	Category of Contact Numbers
PB_SPRAS	☑	☑	SPRAS	LANG	1	0	Language Key
PB_CATEGORY_DESC	☐	☐	Z2812_PB_CATEGO…	CHAR	60	0	Phone Book Category Description

Data – Contact Category

Category	Language	Category Description
HO	E	Home
MO	E	Mobile
OF	E	Office
WO	E	Work

[<< Top](#)

Create Interface Views (CDS)

1. Design / Prototype
2. Create Dictionary Objects
3. **Create Interface View (CDS)**
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

In this step, we will create basic views for our DB table. These views can later be used by other CDS views or any DB fetch.

Note: *This blog will not get into basic steps of installing ADT and connecting backend system to ADT or How to create CDS views. You can refer to below links for the same, in case this is your first-hand experience with ADT & CDS*

- Installing ABAP Development Tools for Eclipse
- Work with Core Data Services
- CDS Annotations

Header Interface View

```
@AbapCatalog.sqlViewName: 'Z2812IPBHEAD'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@ObjectModel:{
  usageType:{
    sizeCategory: #L,
    serviceQuality: #X,
    dataClass: #TRANSACTIONAL
  }
}
@VDM.viewType: #BASIC
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book Header Inerface View'
define view Z2812_I_PB_HEAD
  as select from z2812_pb_d_head as headData {
  //z2812_pb_header
  key pb_head_uuid,
  pb_id,
  pb_owner,
  pb_first_name,
  pb_last_name,
  pb_email_id,
  pb_created_at,
  pb_changed_at
}
```

Item Interface View

```
@AbapCatalog.sqlViewName: 'Z2812IPBITEM'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@ObjectModel:{
  usageType:{
    sizeCategory: #L,
    serviceQuality: #X,
    dataClass: #TRANSACTIONAL
  }
}
@VDM.viewType: #BASIC
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book Item Inerface View'
define view Z2812_I_PB_ITEM
  as select from z2812_pb_d_item as itemData{
  //z2812_pb_item
  key pb_item_uuid,
  pb_head_uuid,
  pb_category,
  pb_telephone,
  pb_created_at,
  pb_changed_at
}
```

Category Interface View

Basic View

```
@AbapCatalog.sqlViewName: 'Z2812IPBCATG'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
```

```
@AccessControl.authorizationCheck: #CHECK
@ObjectModel:{
  usageType: {
    serviceQuality: #B,
    sizeCategory: #S,
    dataClass: #MASTER
  }
}
@VDM.viewType: #BASIC
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book Category Interface View'
define view Z2812_I_PB_CATG as select from z2812_pb_d_catg{
  //z2812_pb_d_catg
  key pb_category,
  key pb_spras,
  pb_category_desc
}
```

Value Help View

```
@AbapCatalog.sqlViewName: 'Z2812IPBCATVH'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@ClientHandling.algorithm: #SESSION_VARIABLE
@ObjectModel:{
  dataCategory: #VALUE_HELP,
  representativeKey: 'PB_CATEGORY',
  usageType: {
    sizeCategory: #S,
    serviceQuality: #B,
    dataClass: #MASTER
  }
}
@VDM.viewType: #BASIC
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book Category Value Help View'
define view Z2812_I_PB_CATG_VH as select from Z2812_I_PB_CATG
{
```

```
  //Z2812_I_PB_CATG
  key pb_category,
  pb_category_desc
} where pb_spras = $session.system_language
```

Text View

```
@AbapCatalog.sqlViewName: 'Z2812IPBCTGTXT'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
@ObjectModel:{
  dataCategory: #TEXT,
  representativeKey: 'PB_CATEGORY',
  usageType: {
    sizeCategory: #S,
    serviceQuality: #B,
    dataClass: #MASTER
  }
}
@VDM.viewType: #BASIC
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book CategoryText View'
define view Z2812_I_PB_CATG_TXT as select from Z2812_I_PB_CATG
{
  //Z2812_I_PB_CATG
  key pb_category,
  pb_category_desc
} where pb_spras = $session.system_language
```

Note: *Value Help and Text view are not categories of views. These are just typed and used for better understanding. Different types of views are Basic, Composite, Consumption, Extension, Structure, Transactional*

[<< Top](#)

Business Object Generation (CDS – Object Model Annotations)

1. Design / Prototype
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. **Business Object Generation (CDS – Object Model Annotations)**
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

First, Item Transactional View was created without any association to header view/node, then Header Transactional View was created with an association to the item view/node. Later, the Association to header was updated in the item view.

Item Transactional View

Draft table name is proposed and system will auto generate the table.

```
@AbapCatalog.sqlViewName: 'Z2812IPBITEMND'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK

-- Business Object Model - Item Data
@ObjectModel:{
  writeActivePersistence: 'Z2812_PB_D_ITEM',
  draftEnabled: true,
  writeDraftPersistence: 'Z2812_PB_D_ITM_D',
  createEnabled: true,
  updateEnabled: true,
  deleteEnabled: true,
  entityChangeStateId: 'PB_CHANGED_AT',
       lifecycle.enqueue.expiryBehavior                 :
#RELATIVE_TO_LAST_CHANGE,
  lifecycle.enqueue.expiryInterval    : 'PT60M',
  usageType: {
    serviceQuality: #D,
    sizeCategory: #L,
    dataClass: #TRANSACTIONAL
  }
}

@VDM.viewType: #TRANSACTIONAL
@EndUserText.label: 'Phone Book Item Transactional - Node
View'
define view Z2812_I_PB_ITEM_ND
   as select from Z2812_I_PB_ITEM as itemData
  -- Category Description
  left outer join Z2812_I_PB_CATG_TXT as _categoryText
                        on _categoryText.pb_category =
itemData.pb_category
```

```
  -- Association to Header
  association [1..1] to Z2812_I_PB_HEAD_ND as _headData
                          on $projection.pb_head_uuid =
_headData.pb_head_uuid
{
  //itemData
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  key pb_item_uuid,
  @ObjectModel.readOnly: true
  pb_head_uuid,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  @ObjectModel.mandatory: true
  itemData.pb_category,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_category_desc,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  @ObjectModel.mandatory: true
  pb_telephone,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_created_at,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_changed_at,
  @ObjectModel.association.type: [#TO_COMPOSITION_PARENT,
                                  #TO_COMPOSITION_ROOT]
  _headData
}
```

Header Transactional View

Object Model Annotations used in View will auto generate Business Object and dependent node such as message, Lock and

```
@AbapCatalog.sqlViewName: 'Z2812IPBHEADND'
@AbapCatalog.compiler.compareFilter: true
@AbapCatalog.preserveKey: true
@AccessControl.authorizationCheck: #CHECK
```

```
-- Business Object Model - Header Data
@ObjectModel:{
  modelCategory: #BUSINESS_OBJECT,
  transactionalProcessingEnabled: true,
  compositionRoot: true,

  semanticKey: ['PB_ID'],
  alternativeKey: [{
    id: 'PB_ID',
    uniqueness: #UNIQUE_IF_NOT_INITIAL,
    element: ['PB_ID']
  }],
  writeActivePersistence: 'Z2812_PB_D_HEAD',
  draftEnabled: true,
  writeDraftPersistence: 'Z2812_PB_D_HDR_D',
  createEnabled: true,
  updateEnabled: true,
  deleteEnabled: true,
  entityChangeStateId: 'PB_CHANGED_AT',
  lifecycle.enqueue.expiryBehavior: #RELATIVE_TO_LAST_CHANGE,
  lifecycle.enqueue.expiryInterval: 'PT60M',
  usageType: {
    serviceQuality: #D,
    sizeCategory:   #L,
    dataClass:      #TRANSACTIONAL
  }
}

@VDM.viewType: #TRANSACTIONAL
@Metadata.ignorePropagatedAnnotations: true
@EndUserText.label: 'Phone Book Head Transactional - Node View
- BO'
define view Z2812_I_PB_HEAD_ND
  as select from Z2812_I_PB_HEAD as headData
  association [0..*] to Z2812_I_PB_ITEM_ND as _itemData
                          on $projection.pb_head_uuid =
_itemData.pb_head_uuid
{
  //headData
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  key pb_head_uuid,
```

```
  @ObjectModel.readOnly: true
  @ObjectModel.mandatory: true
  pb_id,
  @ObjectModel.readOnly: true
  @ObjectModel.mandatory: true
  pb_owner,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_first_name,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_last_name,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_email_id,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_created_at,
  @ObjectModel.readOnly: 'EXTERNAL_CALCULATION'
  pb_changed_at,
  @ObjectModel.association.type: [#TO_COMPOSITION_CHILD]
  _itemData
}
```

For more detail on Object Model Annotations, please refer to SAP Help.

[<< Top](#)

Generated Business Object (BOPF)

---

1. Design / Prototype
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. **Generated Business Object**
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

BOPF?

The Business Object Processing Framework is an ABAP OO-based framework that provides a set of generic services and functionalities to speed up, standardize, and modularize your development. BOPF manages the entire life cycle of your business objects and covers all aspects of your business application development. Instead of expending effort for developing an application infrastructure, the developer can focus on the individual business logic. Using BOPF, you get the whole application infrastructure and integration of various components for free. This allows you to rapidly build applications on a stable and customer-proven infrastructure.

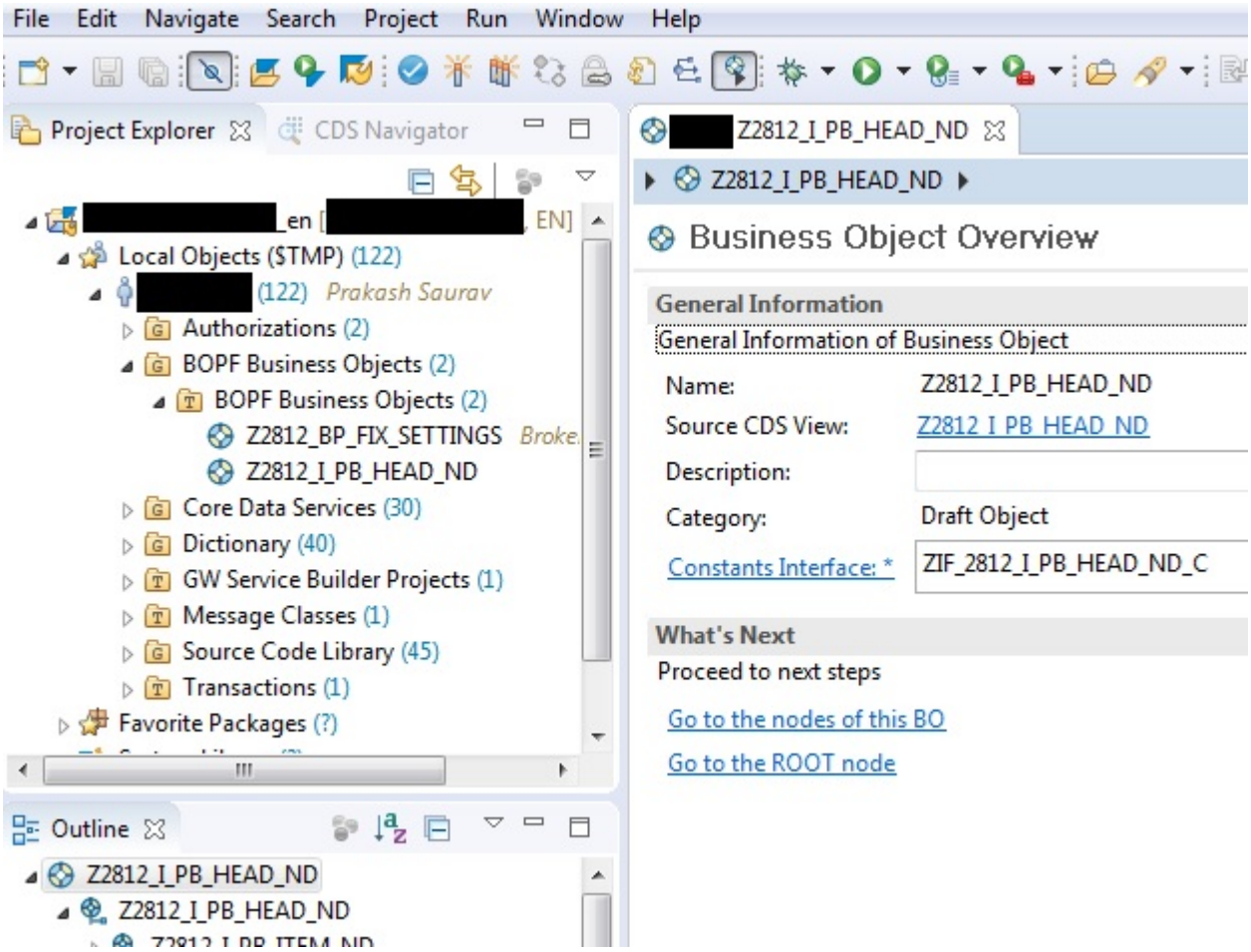
SOURCE: Introduction to Business Object Processing Framework (BOPF)

Refer Source for detailed information on BOPF, it will help you to get an overview of BOPF.

Note: *Transactions to work with BOPF objects in SAPGUI are **BOB**, **BOBF**, **BOBX**. However, Business Object generated using Object Model Annotations in CDS, can only be edited using **ADT** (ABAP Development Tools) in Eclipse.*

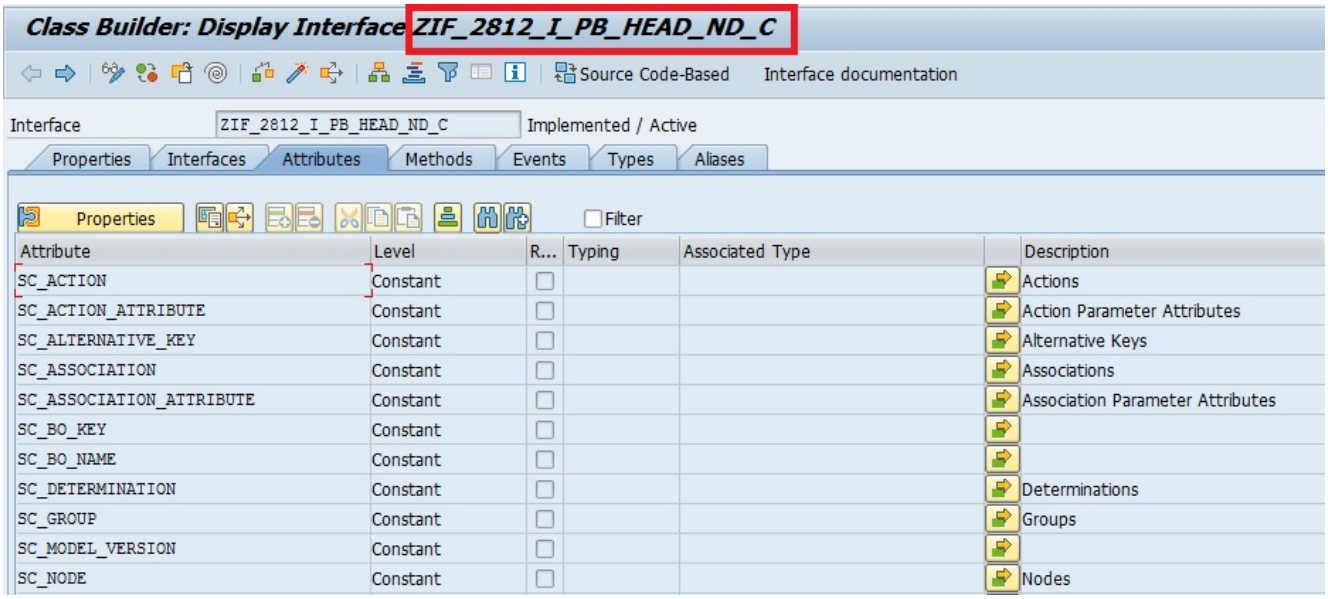
Business Object Initial View

- Name of the generated Business Object is the same as the CDS View.
- Constant Interface is automatically generated
- Go To nodes of this BO => Display Nodes in Hierarchy
- Go to the Root Node => Display Node and it Elements
    - Overview
    - Alternative Keys
    - Properties
    - Associations
    - Actions
    - Determinations
    - Validations
    - Authorizations
- Display Generated Business Object using **BOBF**



Constant Interface

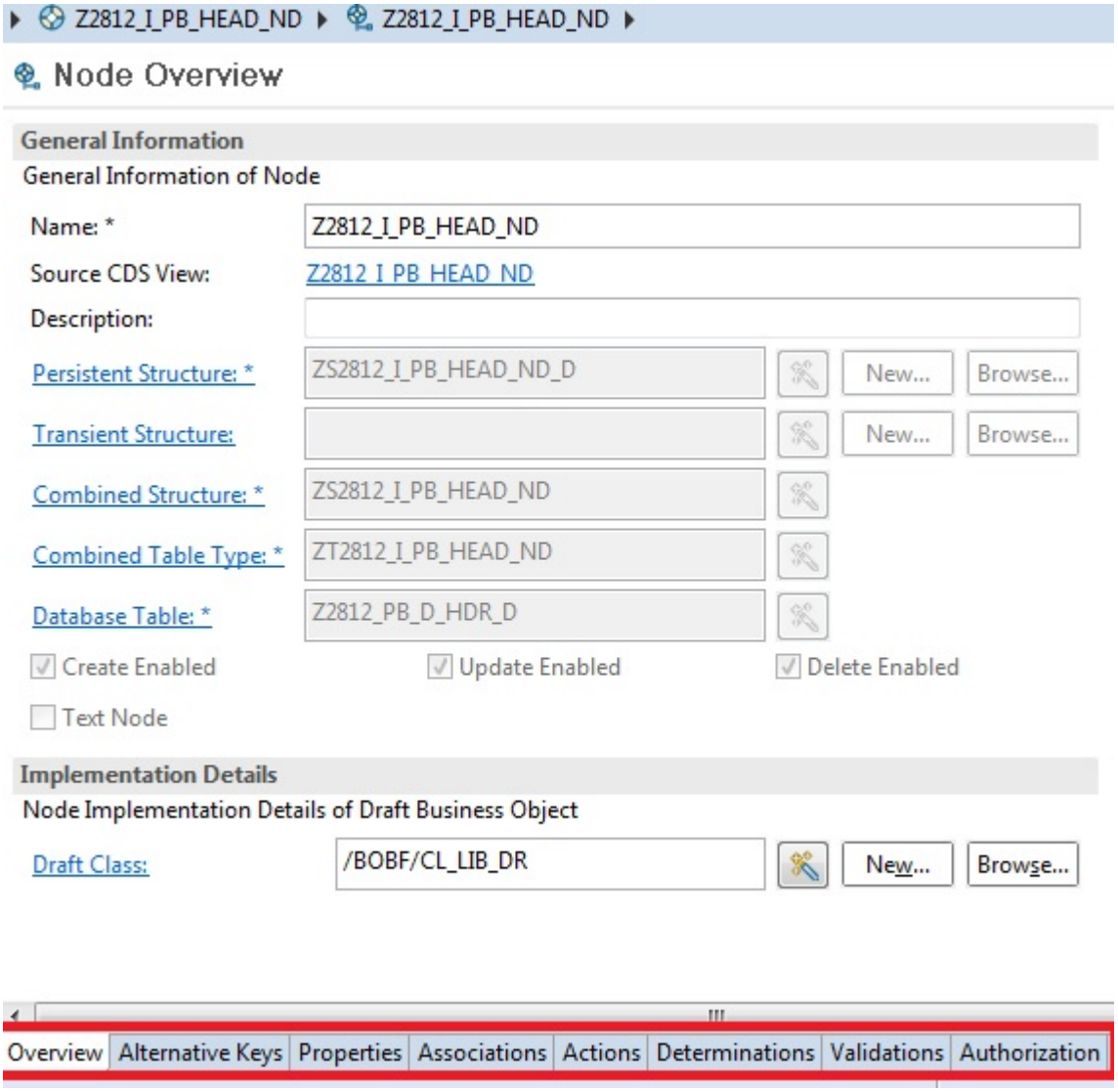
Constant Interface '**ZIF_2812_I_PB_HEAD_ND_C**' is auto-generated. BOPF generates business logic and lots of dependent objects. All of these objects are always referred to using GUID. In SAP GUID is 16 character hexadecimal value, and that is hard to remember to refer objects related to BO. Therefore, the constant interface is generated with proper human readable and understandable name for such objects. To see the values switch to source code based display.



[<< Initial View](#)

Root Node Overview

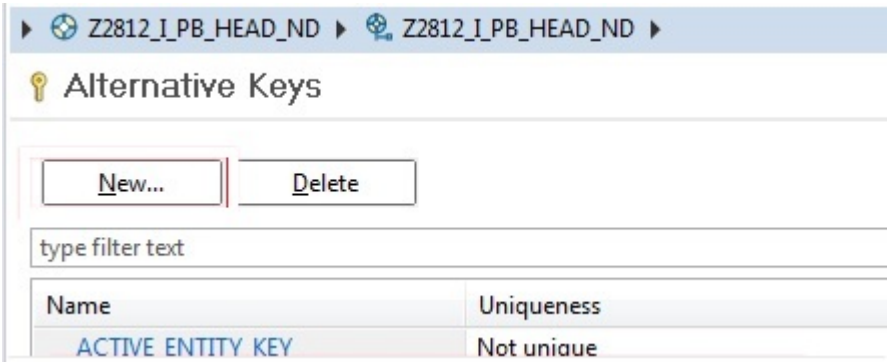
Persistent Structure, Combined Structure (Persistent + Transient Structure) & Combined Table Type are auto-generated. Standard Draft Library Class is assigned for the handling of Drafts.



[<< Initial View](#)

Alternative Keys

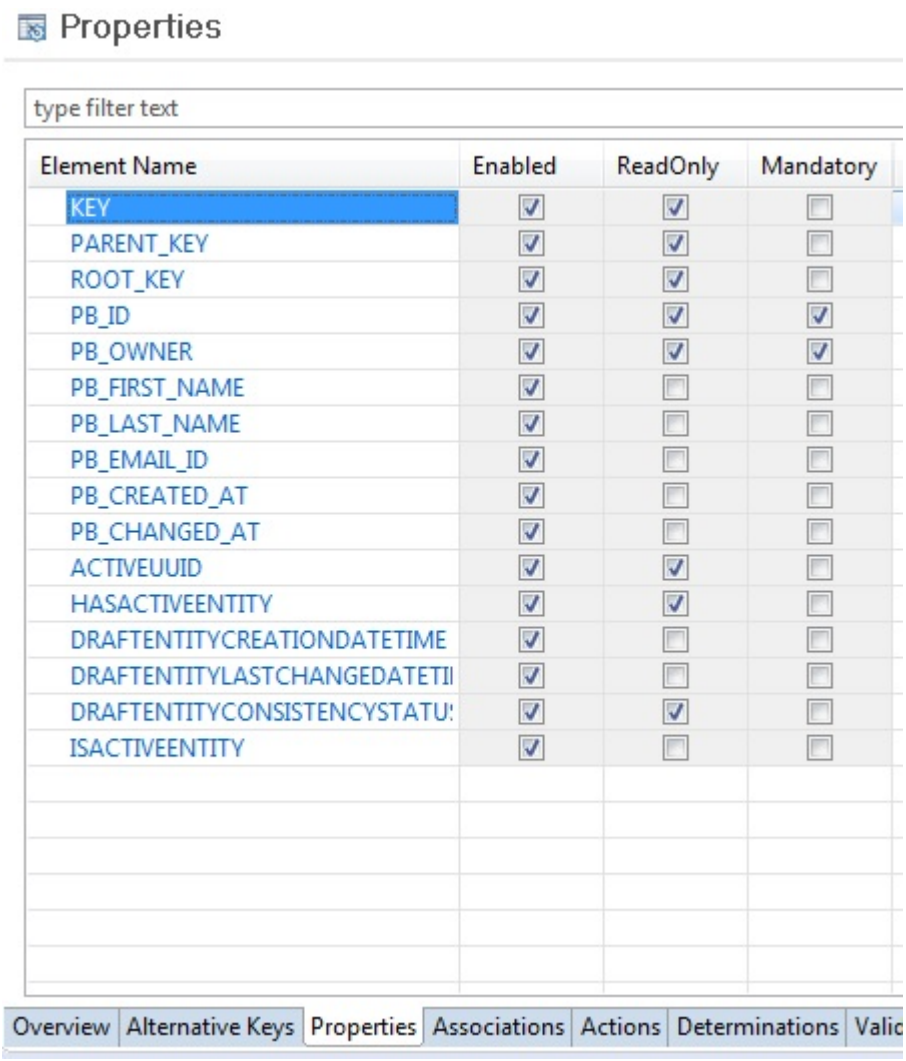
As the name implies, they are an alternate way of finding the instance of the node. It is more meaningful and related to business data, for example in this scenario Phone Book ID can be created and used as Alternate Key.



[<< Initial View]

Properties

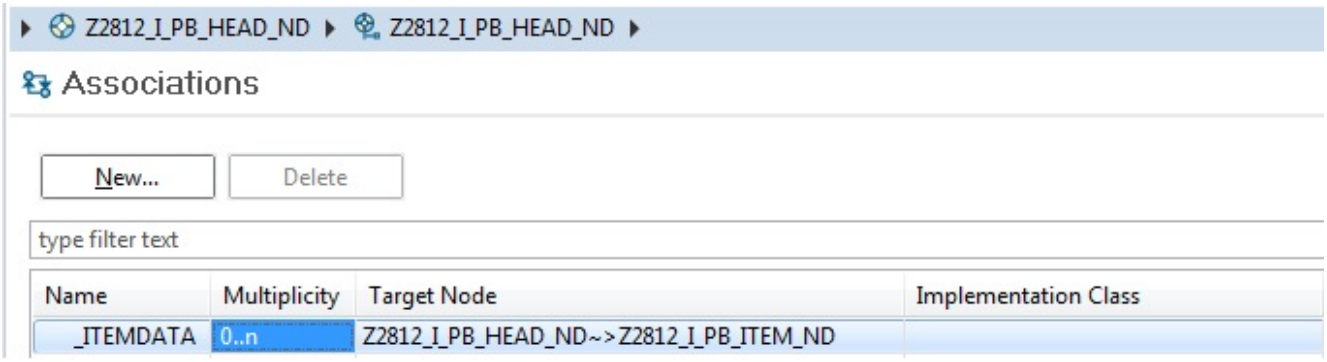
Initially, the properties of fields are set as instructed by annotations. properties can be set for Node, Node Attributes, Associations, Actions etc. Example of properties that can be set are Enabled, Read Only, Mandatory, Create Enabled, Update Enabled, Delete Enabled.



[<< Initial View](#)

Associations

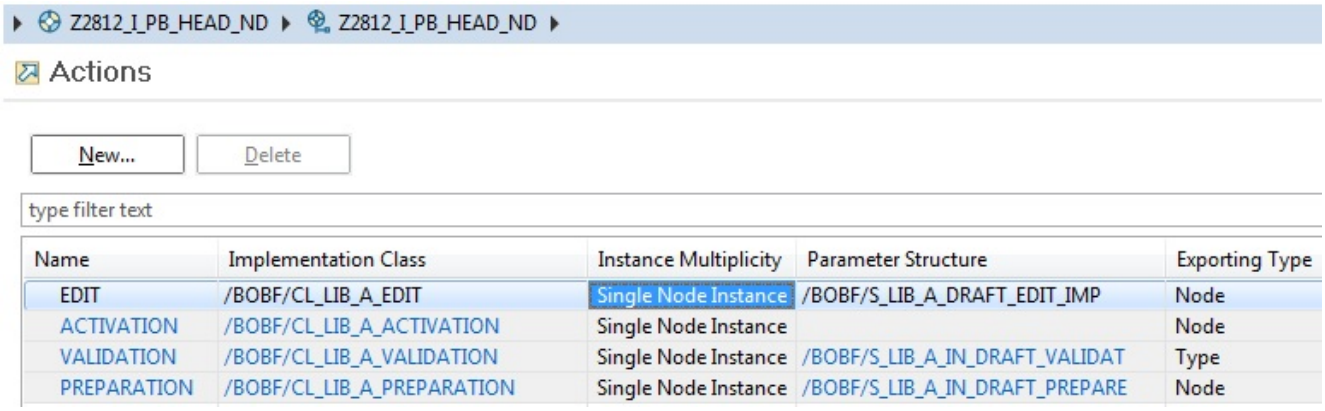
Association to Item Data is generated as per CDS annotation.



[<< Initial View]

Actions

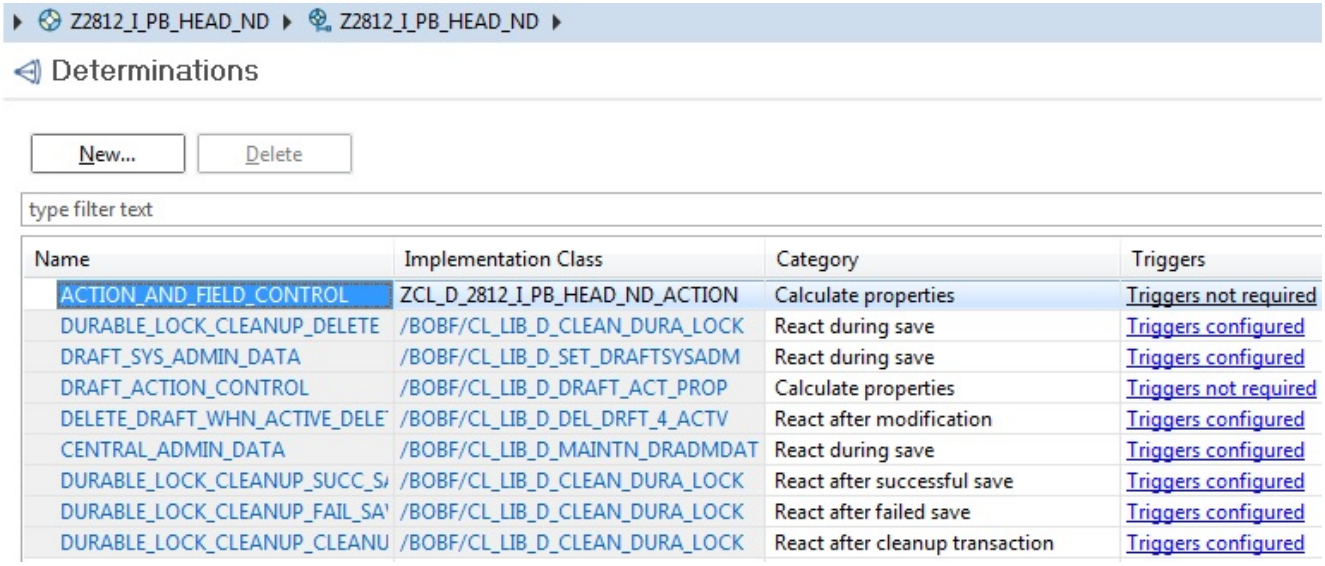
Action carries out the application of business logic. There are many standard actions generated for CRUD and Draft handling. Action has to be triggered explicitly.



[<< Initial View]

Determinations

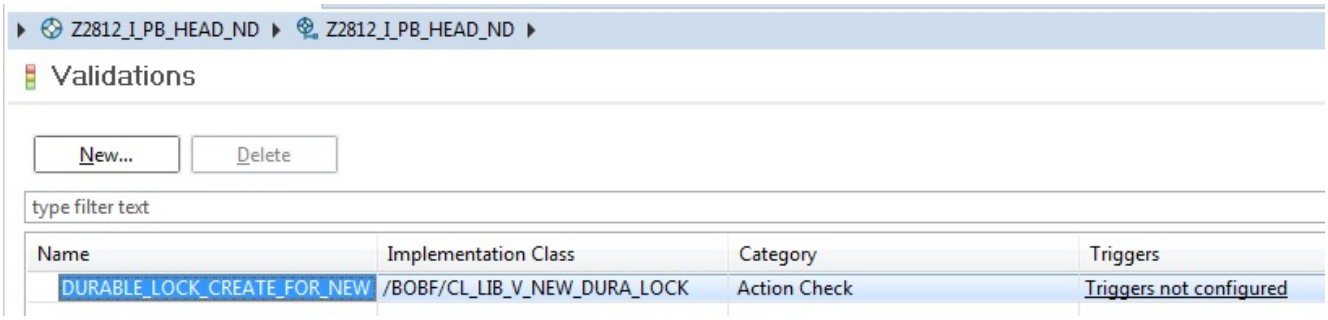
Determinations are tied to the event and are triggered implicitly. You cant call a determination directly. Many standard determinations are auto-generated for Drafts, Locks, Field Control etc. Example, in case, if Transient Structure is used then determinations can be used to update transient data. However, it can also be used to update persistent data as a result of the change of values of other attributes.



[<< Initial View]

Validations

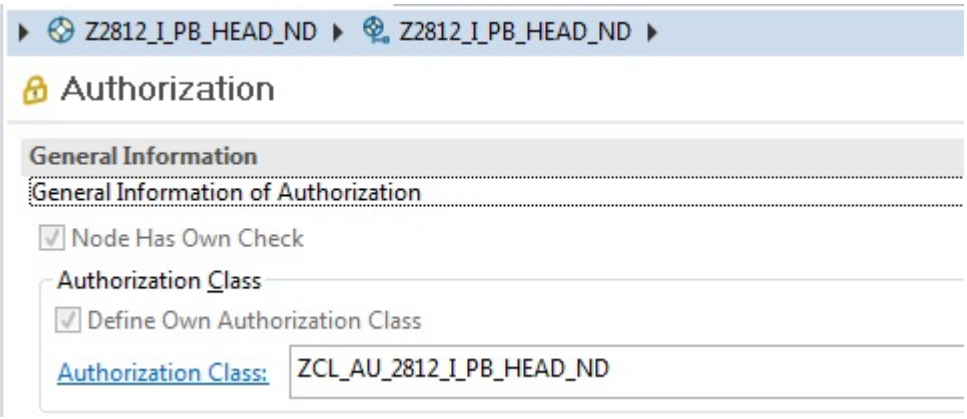
Validation is checks triggered to validate the Node.
Action validations are pre-checks to continue with the triggered action.
Consistency validations do node consistency checks and can stop CRUD operations based on configuration or logic.



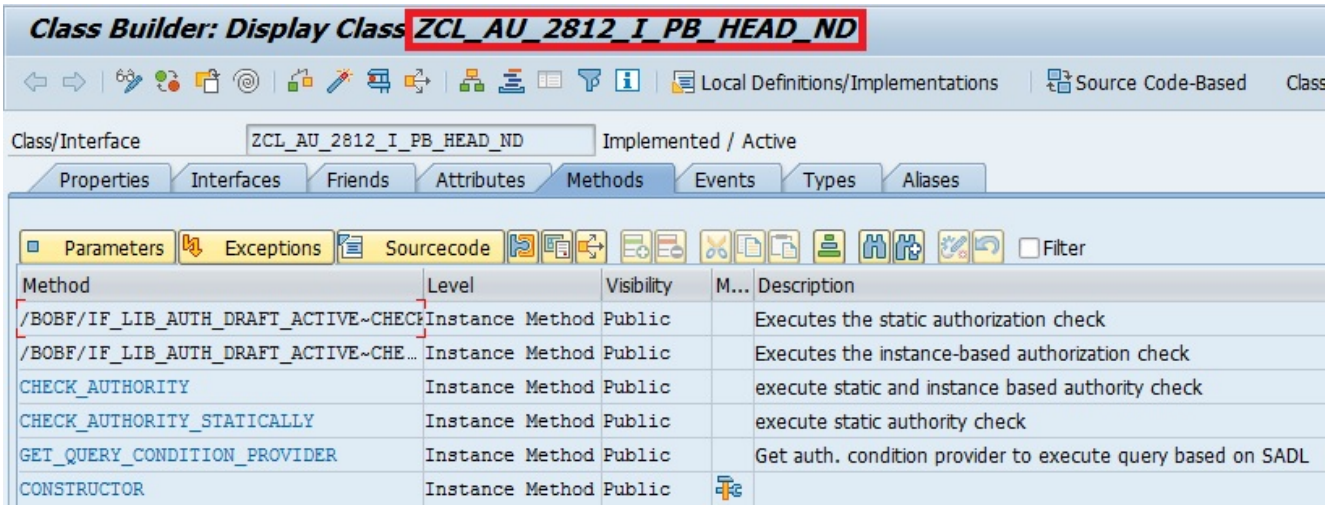
[<< Initial View]

Authorizations

For handling authorization framework generates Authorization element and assign Authorization Class to it.



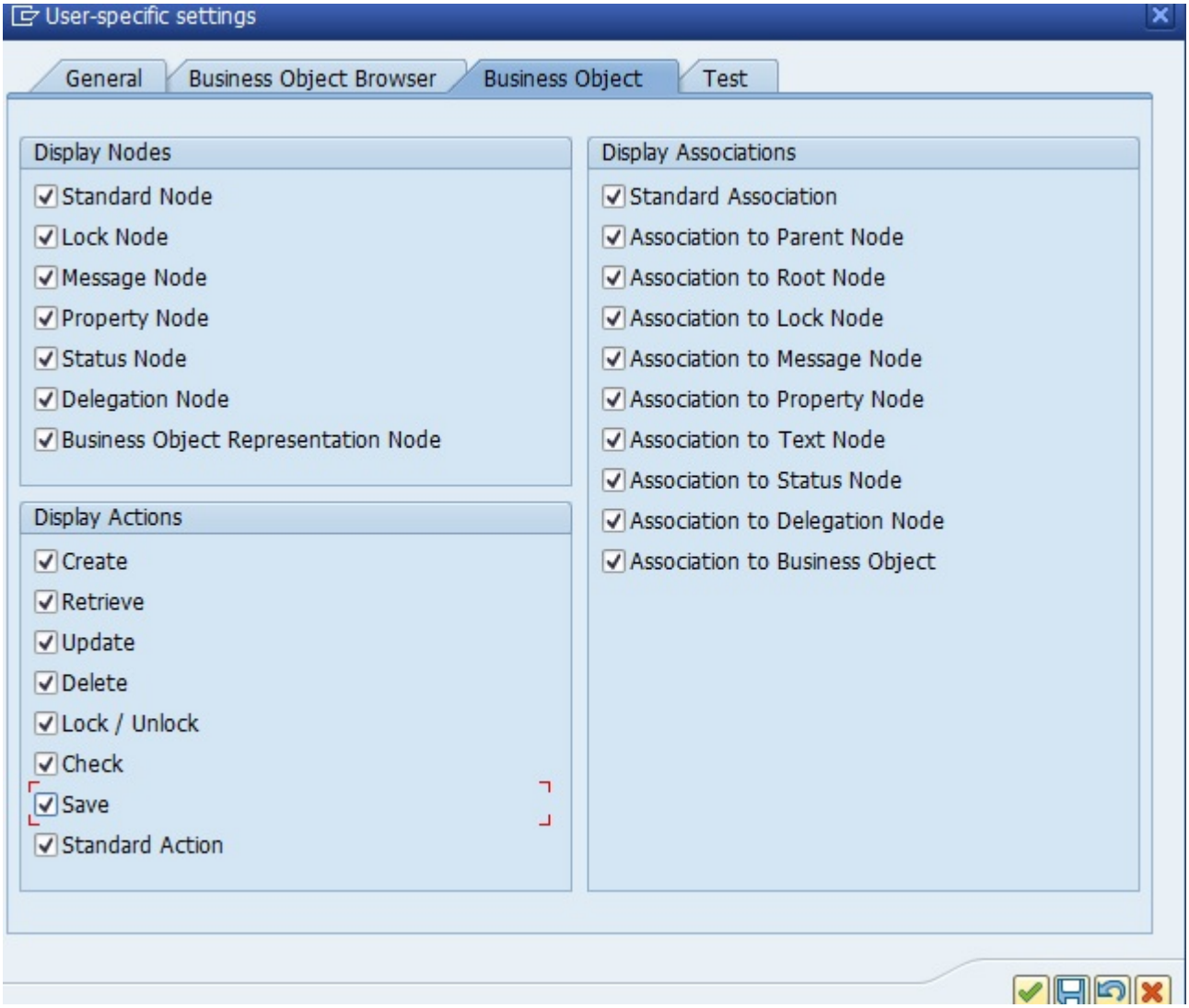
Based on the requirement you can use method ‘CHECK_STATIC_AUTHORITY’ or ‘CHECK_INSTANCE_AUTHORITY’



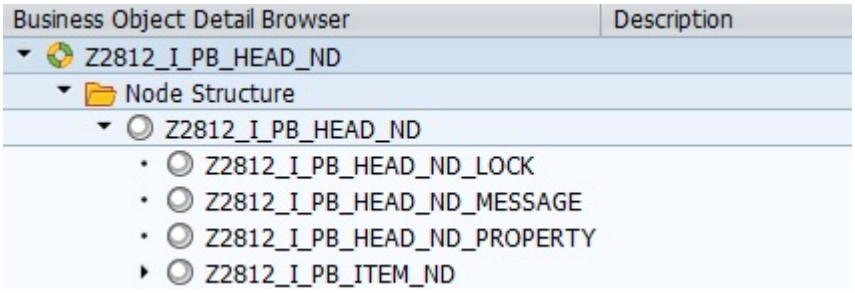
[<< Initial View]

Display Generated Business Object using BOBF

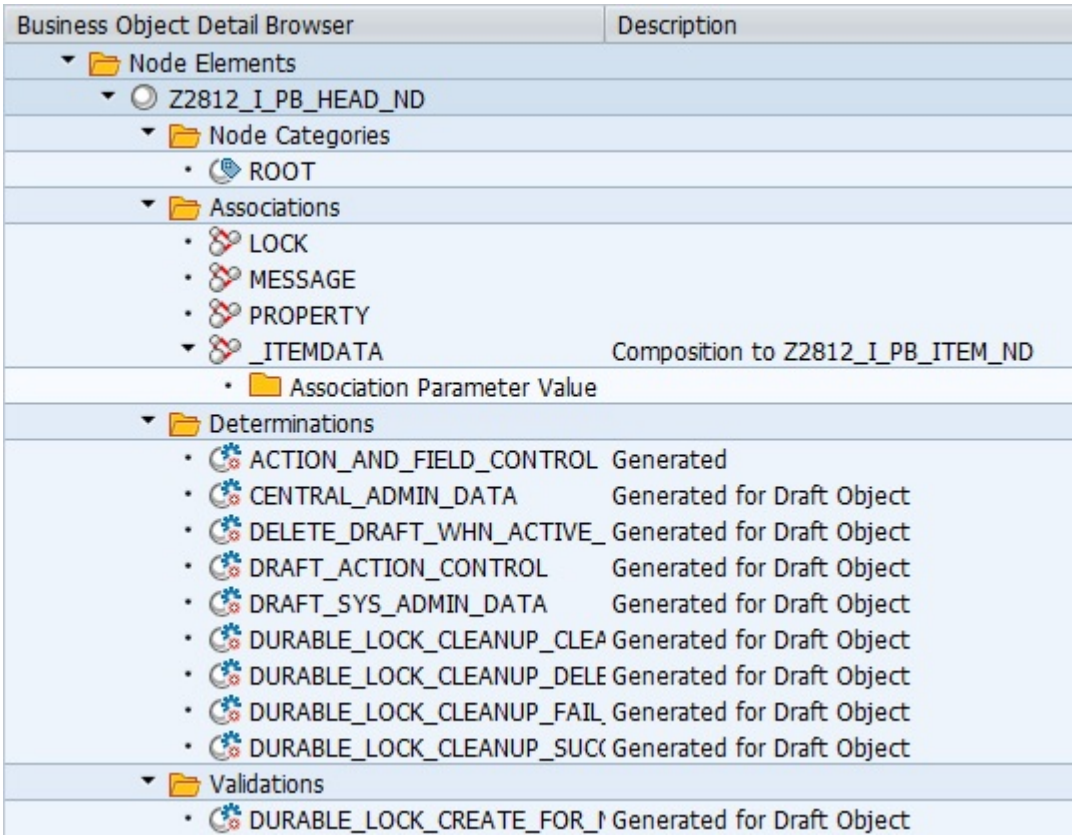
Many associations to other pre-existing BO’s such as Lock, Message and Property are created. They are not displayed by default. We need to change the User Settings in BOBF transaction for Business Object, in order to display all.



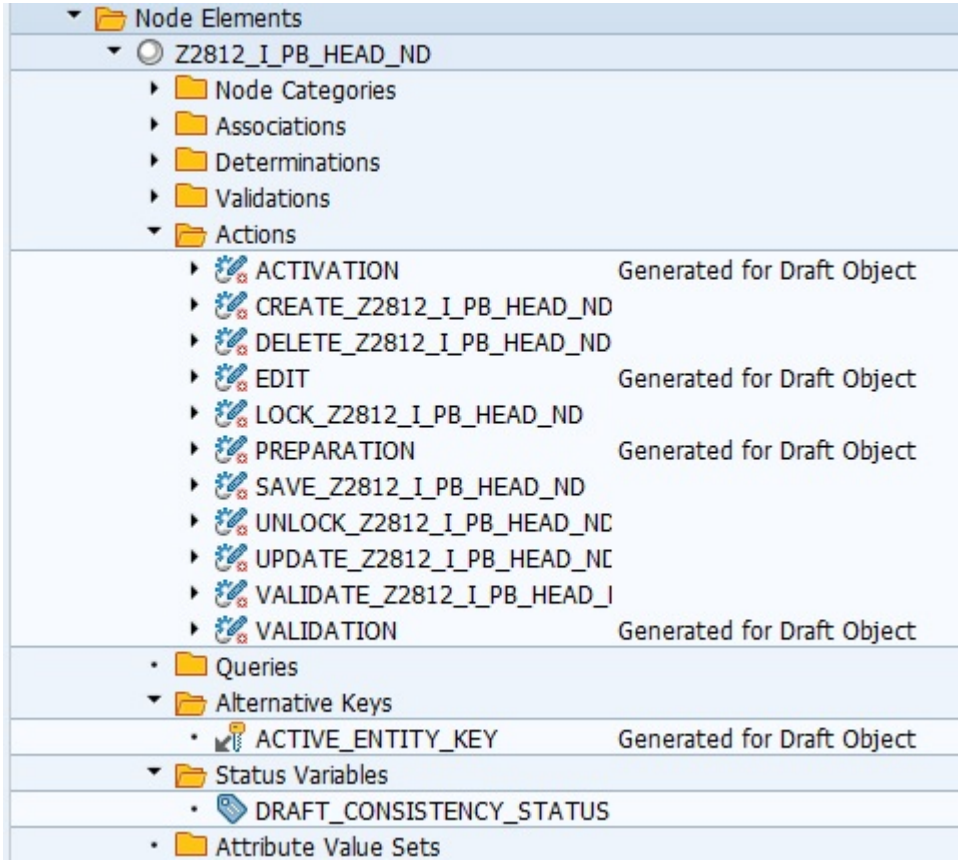
Node Structure View



Node Elements



Node Elements Image 1



Node Elements Image 2

[<< Initial View]

[<< Top]

BOPF Development

---

1. Design / Prototype
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. **BOPF – Development**
    - **Determination**
    - **Validation**
    - **Action**
    - **Alternative Key**
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

As per Phone Book Application requirement, we will develop

following:

1. Determinations
    - GENERATE_PBID: Generate Phone Book ID
    - UPDATE_H_ADMIN: Update Creation and Change timestamp.
    - UPDATE_I_ADMIN: Update Creation and Change timestamp.
2. Validations
    - CHECK_EMAIL: Validate E-Mail ID
    - CHECK_TELENO: Validate Telephone Number
3. Actions
    - COPY: Copy or Duplicate the selected contact
4. Alternative Keys
    - PB_ID: Phone Book ID as alternate key

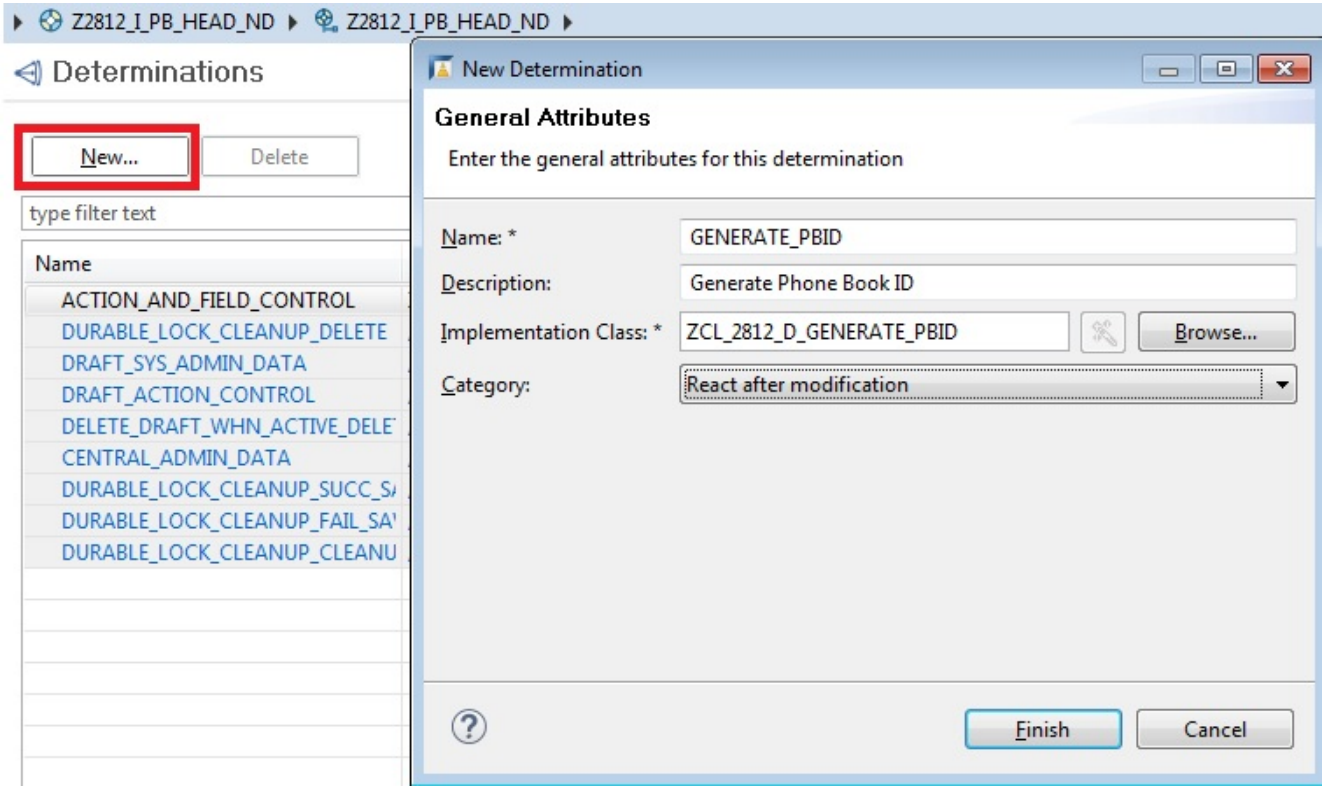
Determinations

Generate Phone Book ID

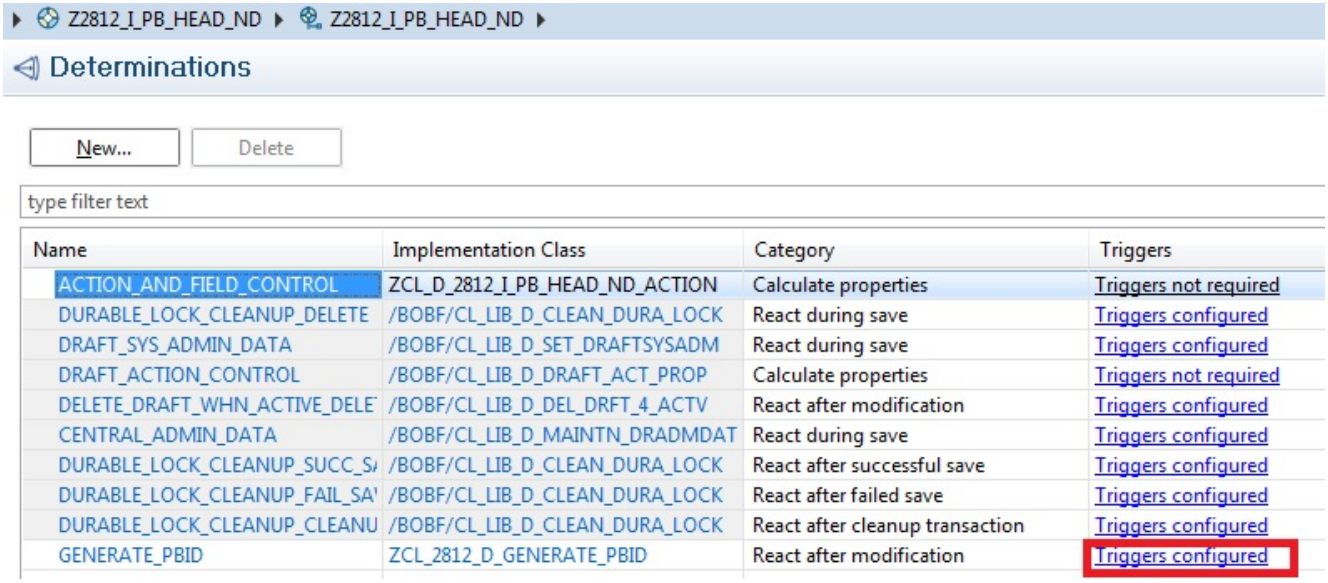
This determination will generate next Phone Book ID in series and update the same to buffer.

Step 1: Navigate to *Business Object >> Go To Root Node >> Select Determinations Tab at the bottom or Click on Determinations Link on the Side*.

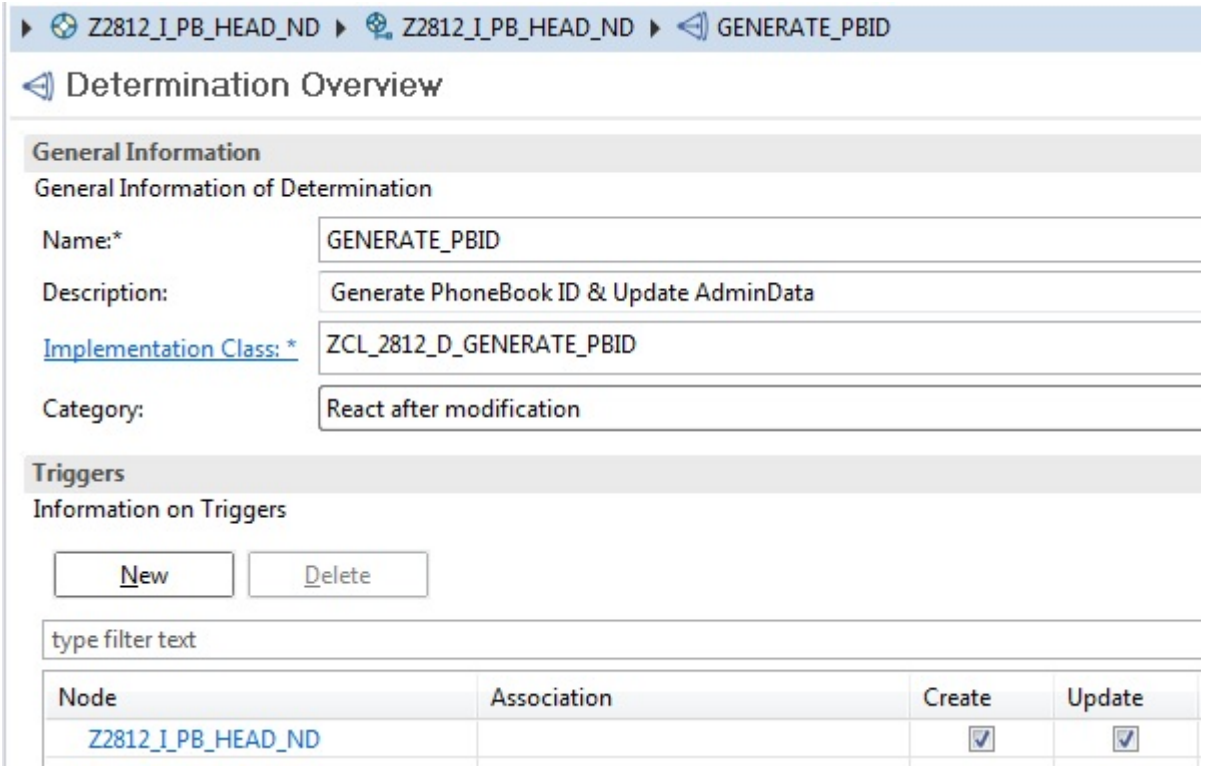
1. Once you are in Determinations Tab then click on **New** to create a determination.
2. Based on Name system will propose Implementation Class, but you can change the class name or re-generate the proposal.
3. Click on Finish
4. Save and Activate



Step 2: Click on **Triggers Configured** or you can double click on the new determination from **Outline View** or you can select the new determination from **Node**, to further configure it.

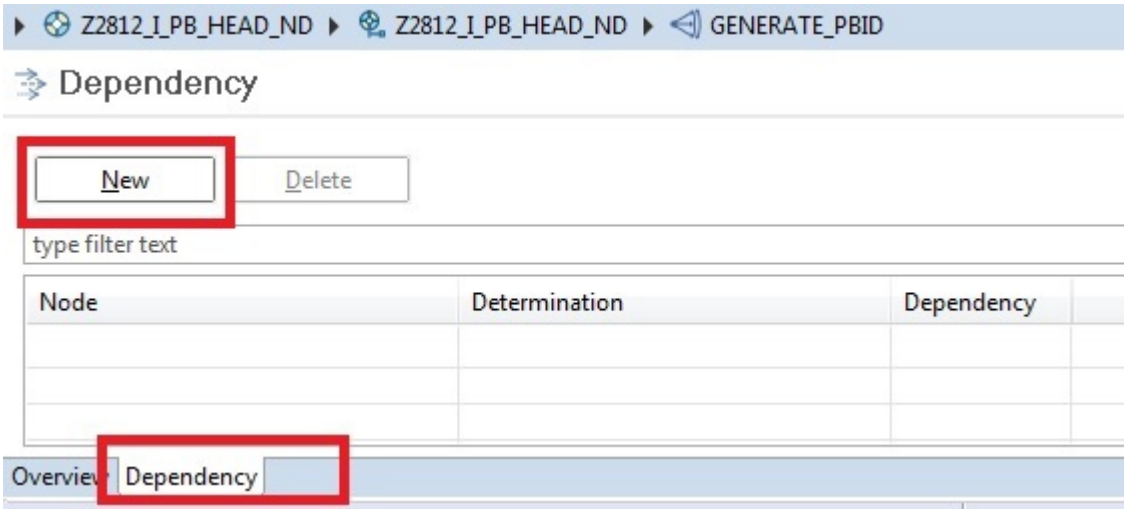


Step 3: In determination configuration, you can see that determination will be executed **After Modification,** it is associated with **Header Node**, and is only triggered for **Create and Update** action. As there is no need to generate new Phone Book ID for Delete action.

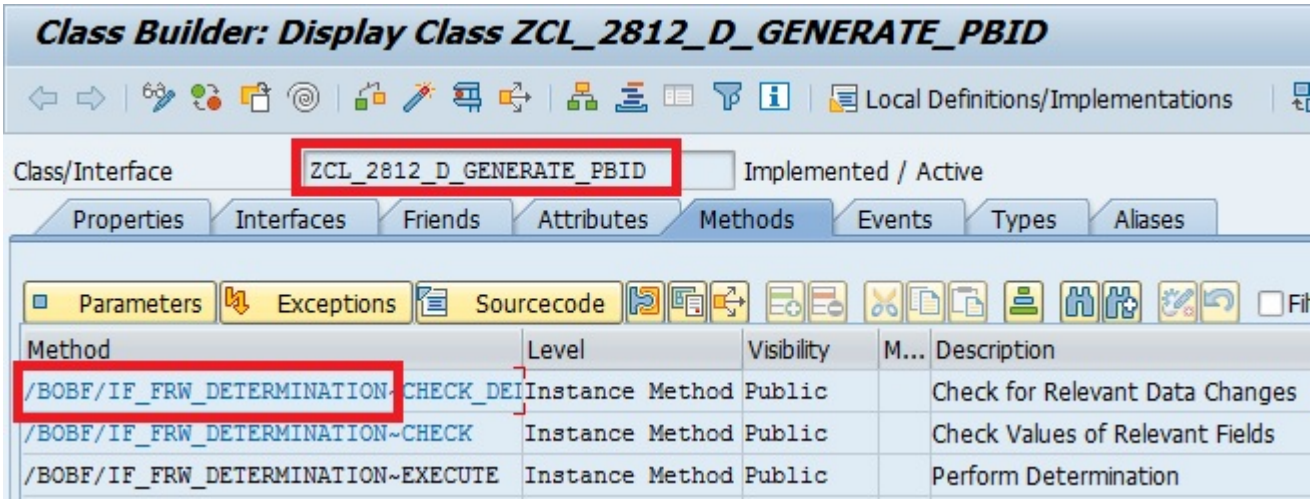


Step 4: If it is required, you can mention **dependencies** in

'Dependency' tab of determination configuration view. This will ensure that dependent determinations will be executed before this determination. Other than this there no control on the sequence of determination execution. *Best practice suggests that there should not be any inter-dependent determinations.*



Step 5: As soon as the determination is activated, *the implementation class is auto-generated* with the necessary interface. Navigate to the class.



Step 6: Implement the business logic for the Determination in the method **Execute**.

```
*---------------------------------------------------------------------*
*   Method  Name                                                      :
/BOBF/IF_FRW_DETERMINATION~EXECUTE
* Method description         : Generate Phone Book ID
*---------------------------------------------------------------------*
  METHOD /bobf/if_frw_determination~execute.
    " Internal tab for Header Data
    " Created using reference to Generated Table Type
    DATA(lt_head) = VALUE zt2812_i_pb_head_nd( ).

    " Get Phone Book Head Data
    io_read->retrieve(
      EXPORTING
        iv_node                = is_ctx-node_key " Node Name
        it_key                 = it_key          " Key Table
      IMPORTING
         et_data                   = lt_head         " Data
Return Struct.
    ).

    " For Each Node Instance
    LOOP AT lt_head REFERENCE INTO DATA(lr_head).
      IF lr_head->pb_id IS INITIAL. " Phone Book ID is Initial
        " Generate Phone Book ID
        CALL FUNCTION 'NUMBER_GET_NEXT'
          EXPORTING
                                nr_range_nr               =
zif_2812_pb_global=>gc_nr_pb_intrvl
            object           = zif_2812_pb_global=>gc_nr_pb_id
          IMPORTING
            number           = lr_head->pb_id
          EXCEPTIONS
            interval_not_found      = 1
            number_range_not_intern = 2
            object_not_found        = 3
            quantity_is_0           = 4
            quantity_is_not_1       = 5
            interval_overflow       = 6
            buffer_overflow         = 7
```

```
          OTHERS                  = 8.
        IF sy-subrc EQ 0.
          lr_head->pb_owner = sy-uname.
          GET TIME STAMP FIELD DATA(lv_timestamp).
          lr_head->pb_created_at = lv_timestamp.
          lr_head->pb_changed_at = lv_timestamp.
        ELSE.
          " Phone Book ID is marked as mandatory
          " Framework will throw error if it is not supplied
        ENDIF.

        io_modify->update(
          EXPORTING
            iv_node           = is_ctx-node_key       " Node
            iv_key            = lr_head->key          " Key
            is_data           = lr_head               " Data
        ).
      ENDIF.
    ENDLOOP.
  ENDMETHOD.
```

Update Header Admin Data

This determination will update ‘Created At’ and ‘Changed At’ timestamp.

Using the above steps create determination **UPDATE_H_ADMIN**, it will be triggered ‘After Modify’, it is associated with **Header Node**, it is enabled for action **Create** and **Update**.

Business Logic

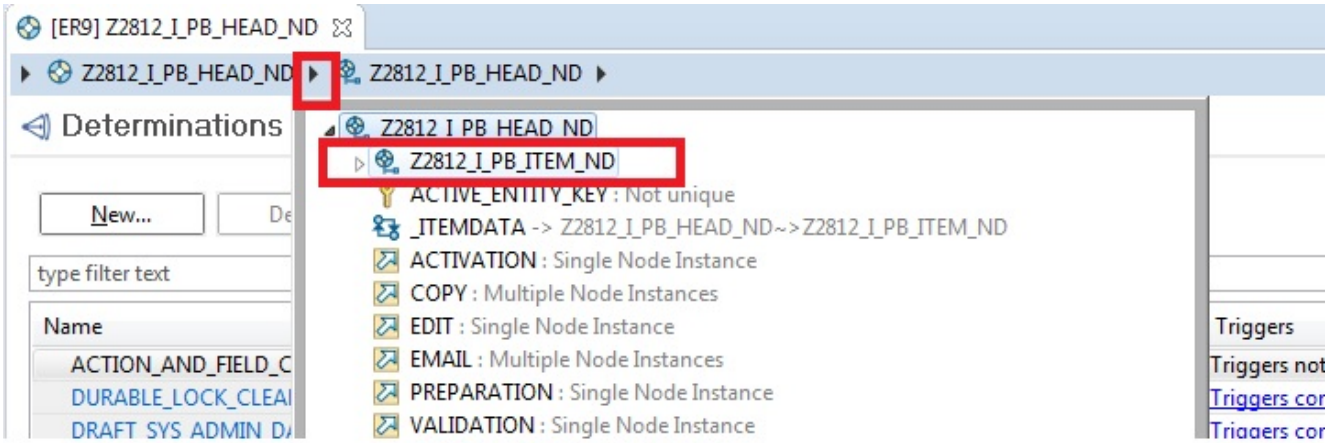
```
*---------------------------------------------------------------------*
```

```
*   Method   Name                                                  :
/BOBF/IF_FRW_DETERMINATION~EXECUTE
* Method description         : Update Admin Head Data
*-------------------------------------------------------------------
---------*
  METHOD /bobf/if_frw_determination~execute.
    " Internal tab for Header Data
    " Created using reference to Generated Table Type
    DATA(lt_head) = VALUE zt2812_i_pb_head_nd( ).

    " Get Phone Book Head Data
    io_read->retrieve(
      EXPORTING
        iv_node                = is_ctx-node_key " Node Name
        it_key                 = it_key          " Key Table
      IMPORTING
        et_data                 = lt_head         " Data Ret.
Struct.
    ).

    LOOP AT lt_head REFERENCE INTO DATA(lr_head).
      GET TIME STAMP FIELD DATA(lv_timestamp).
      lr_head->pb_changed_at = lv_timestamp.

      " Update Data to Node
      io_modify->update(
        EXPORTING
          iv_node           = is_ctx-node_key       " Node
          iv_key            = lr_head->key          " Key
          is_data           = lr_head               " Data
      ).
    ENDLOOP.
  ENDMETHOD.
```

Update Item Admin Data

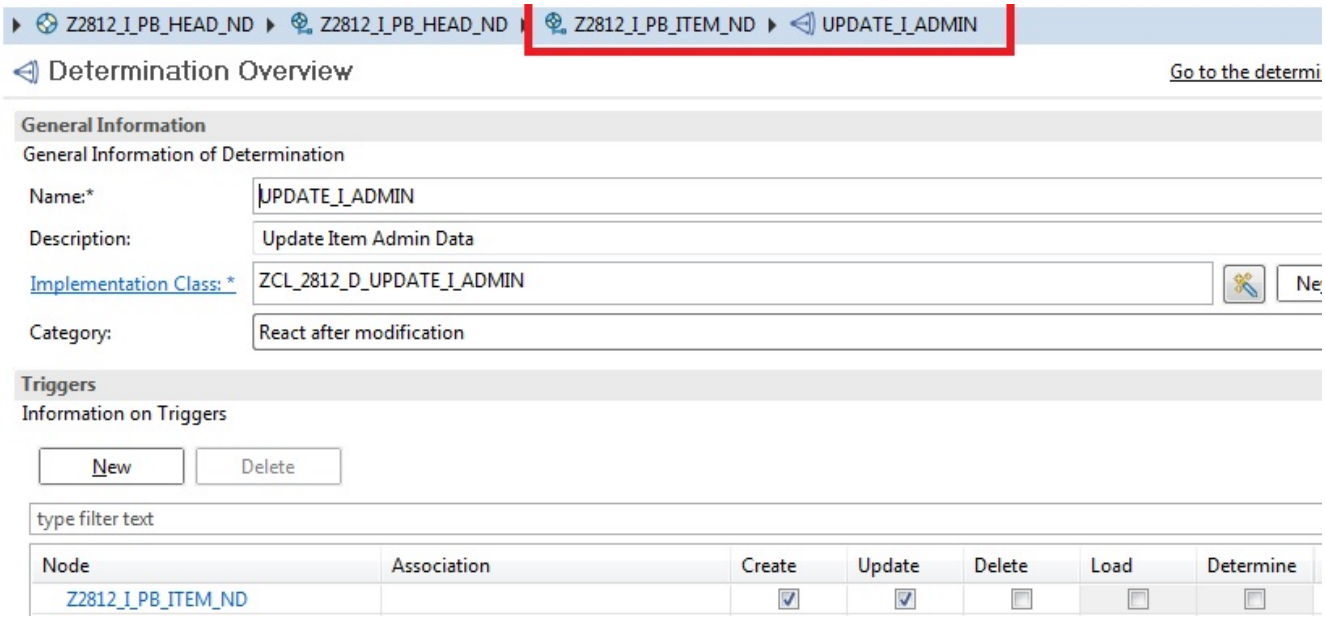
This determination will update ‘Created At’ and ‘Changed At’ timestamp.

Step 1: Navigate to *Business Object >> Go To Root Node >>Select Item Node*.



Step 2: *Select Determinations Tab at the bottom or Click on Determinations Link on Side*

Using remaining steps from GENERATE_PBID determination, create determination **UPDATE_I_ADMIN**, it will be triggered 'After Modify', it is associated with **Header Node**, it is enabled for action **Create** and **Update**.



Business Logic

```
*---------------------------------------------------------------
---------*
*   Method   Name                                              :
/BOBF/IF_FRW_DETERMINATION~EXECUTE
* Method description         : Update Admin Item Data
*---------------------------------------------------------------
---------*
  METHOD /bobf/if_frw_determination~execute.

    " Internal tab for Item Data
    " Created using reference to Generated Table Type
    DATA(lt_item) = VALUE zt2812_i_pb_item_nd( ).

    " Get Item Data
    io_read->retrieve(
      EXPORTING
        iv_node                 = is_ctx-node_key  " Node Name
        it_key                  = it_key           " Key Table
      IMPORTING
        et_data                  = lt_item          " Data Ret.
Struct.
    ).

    " For Each Node Instance
    LOOP AT lt_item REFERENCE INTO DATA(lr_item) .
      GET TIME STAMP FIELD DATA(lv_timstmp).
      IF lr_item->pb_created_at IS INITIAL.
         lr_item->pb_created_at = lv_timstmp.       " Created
At
      ENDIF.


       lr_item->pb_changed_at = lv_timstmp.          " Last
Changed At

      " Update Data to Node
      io_modify->update(
        EXPORTING
```

```
          iv_node           = is_ctx-node_key      " Node
          iv_key            = lr_item->key         " Key
          is_data           = lr_item              " Data
      ).

    ENDLOOP.
  ENDMETHOD.
```

[<< Development List]

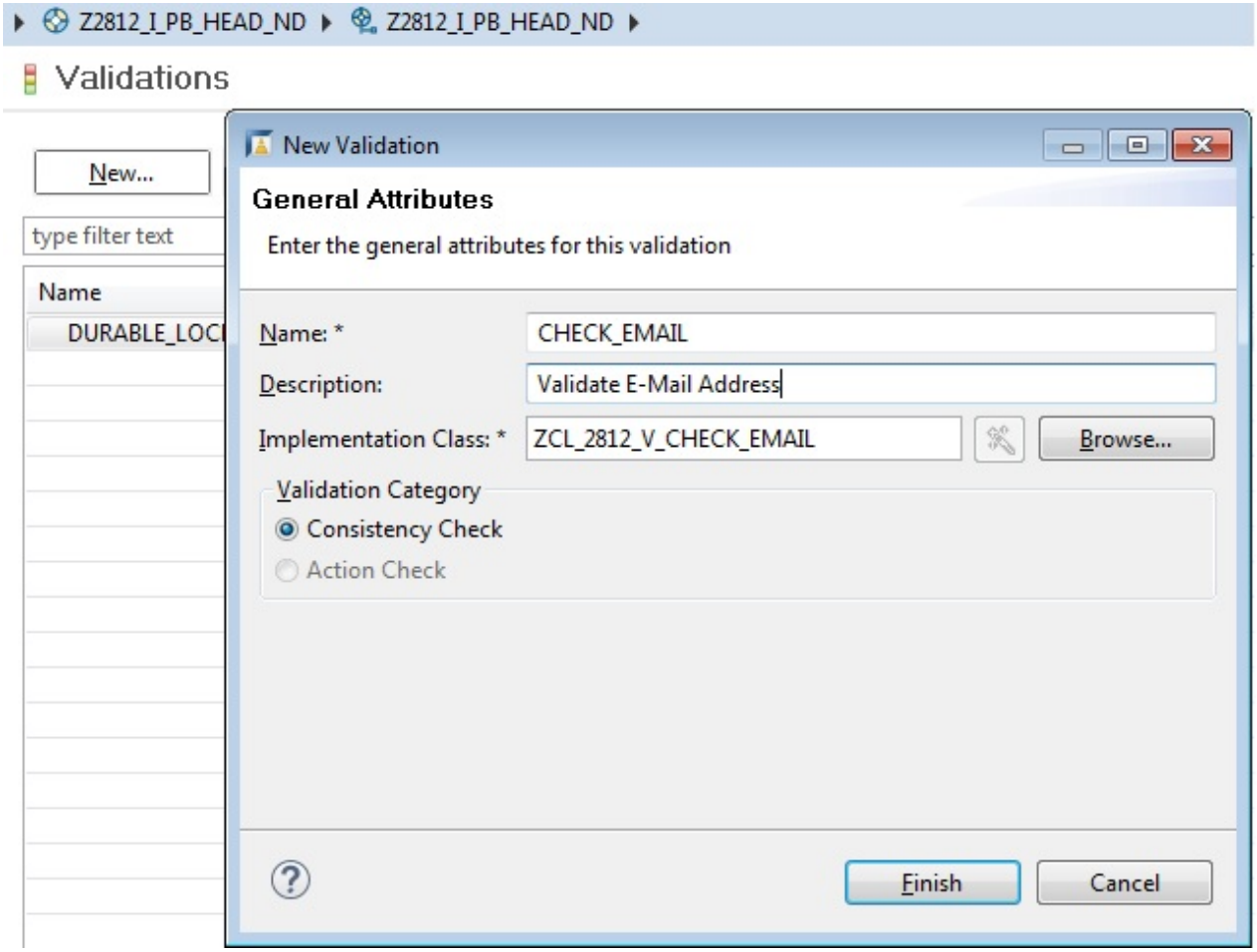
Validations

CHECK_EMAIL

This will validate the E-Mail address

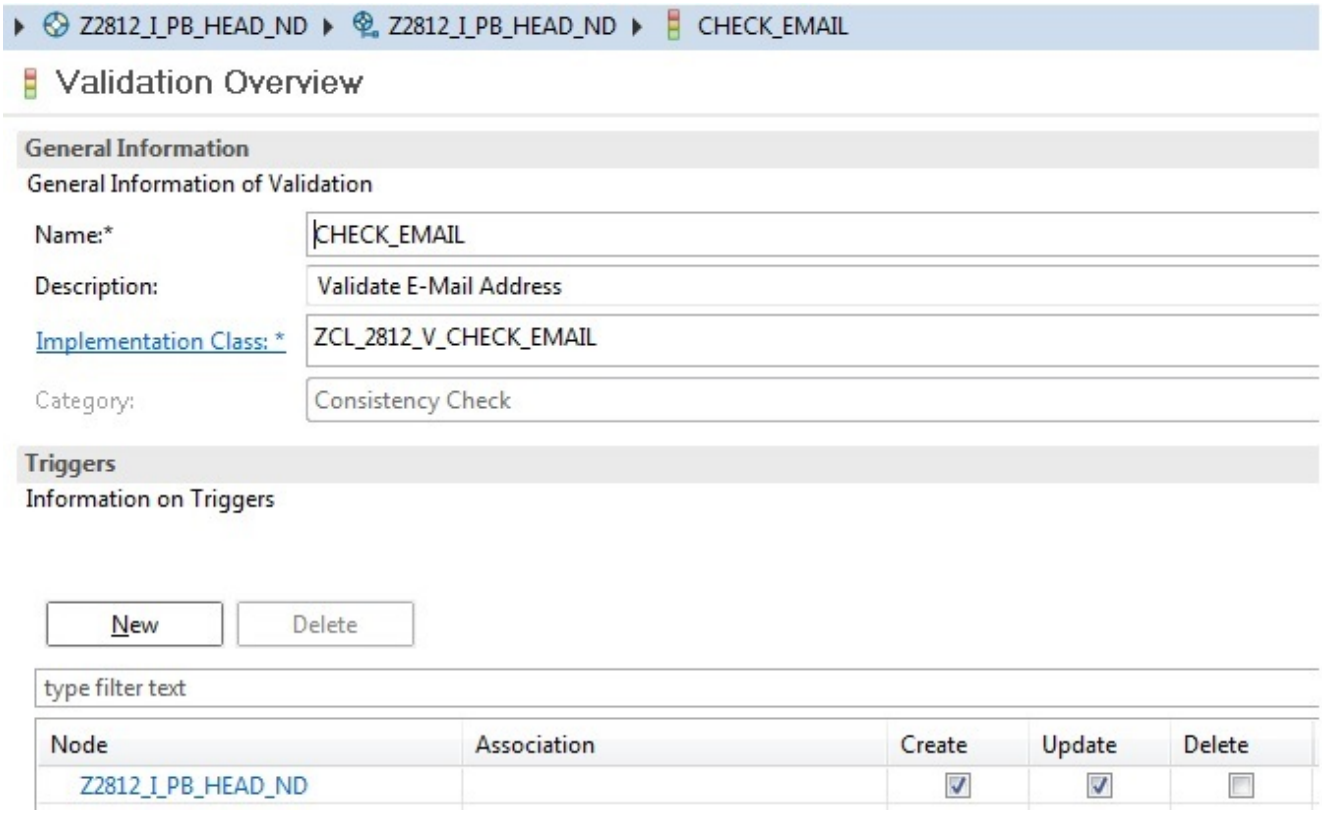
Step 1: Navigate to *Business Object >> Go To Root Node >> Select Validations Tab at the bottom or Click on Validations Link on Side*.

1. Once you are in Validations Tab then click on **New** to create a validation.
2. Based on Name system will propose Implementation Class, but you can change the class name or re-generate the proposal.
3. Click on Finish
4. Save and Activate

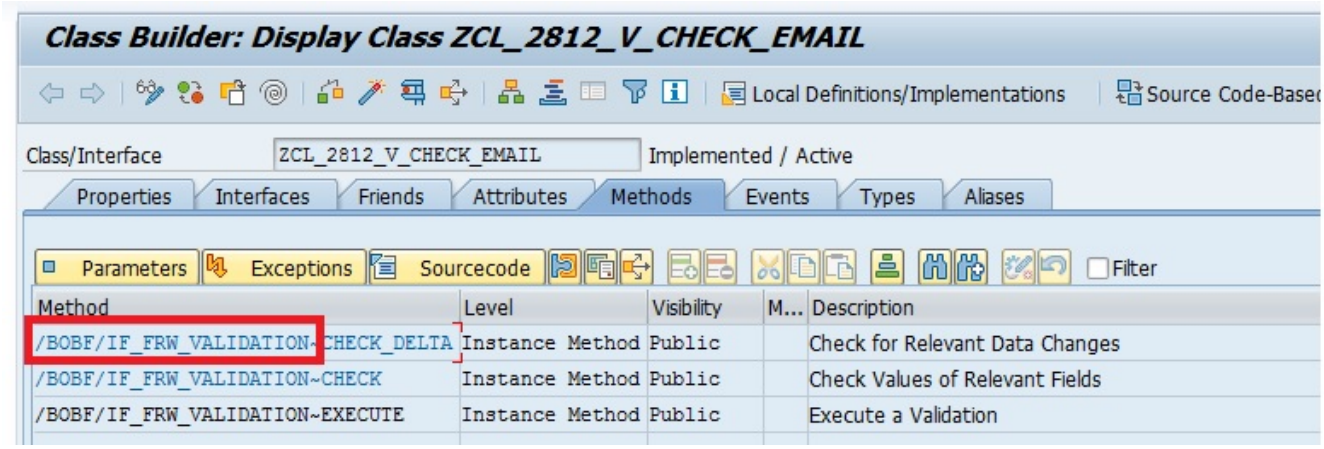


Step 2: Click on **Triggers Configured** or you can double click on the new validation from **Outline View** or you can select the new validation from **Node**, to further configure it.

In the validation configuration, you can see that validation is associated with consistency check, **Header Node**, and is only triggered for **Create and Update** action. As there is no need to validate on the Delete action.



Step 3: As soon as the validation is activated, *the implementation class is auto-generated* with the necessary interface. Navigate to the class.



Step 4: Implement the business logic for the Validation in the method **Execute**.

*----------

```
---------*
* Method Name               : /BOBF/IF_FRW_VALIDATION~EXECUTE
* Method description        : Validate E-Mail Adress
*-------------------------------------------------------------
---------*
  METHOD /bobf/if_frw_validation~execute.
    " Internal tab for Header Data
    " Created using reference to Generated Table Type
    DATA(lt_head) = VALUE zt2812_i_pb_head_nd( ).

    " Get Phone Book Head Data
    io_read->retrieve(
      EXPORTING
        iv_node                = is_ctx-node_key " Node Name
        it_key                 = it_key          " Key Table
      IMPORTING
         et_data                  = lt_head         " Data
Return Structure
    ).

    " Instantiate Message Object
    IF eo_message IS NOT BOUND.
      eo_message = /bobf/cl_frw_factory=>get_message( ).
    ENDIF.

    LOOP AT lt_head REFERENCE INTO DATA(lr_head).
      " Validate
      me->validate_email(
        EXPORTING
          iv_emailid = lr_head->pb_email_id  " Email Address
        IMPORTING
           es_msg     = DATA(ls_msg)          " Structure of
message
      ).

      " Add Message
      IF NOT ls_msg IS INITIAL.
        eo_message->add_message(
          EXPORTING
            is_msg       = ls_msg            " Message
            iv_node      = is_ctx-node_key   " Node
```

```
            iv_key       = lr_head->key      " Instance key
          ).
        ENDIF.
      ENDLOOP.
    ENDMETHOD.
```

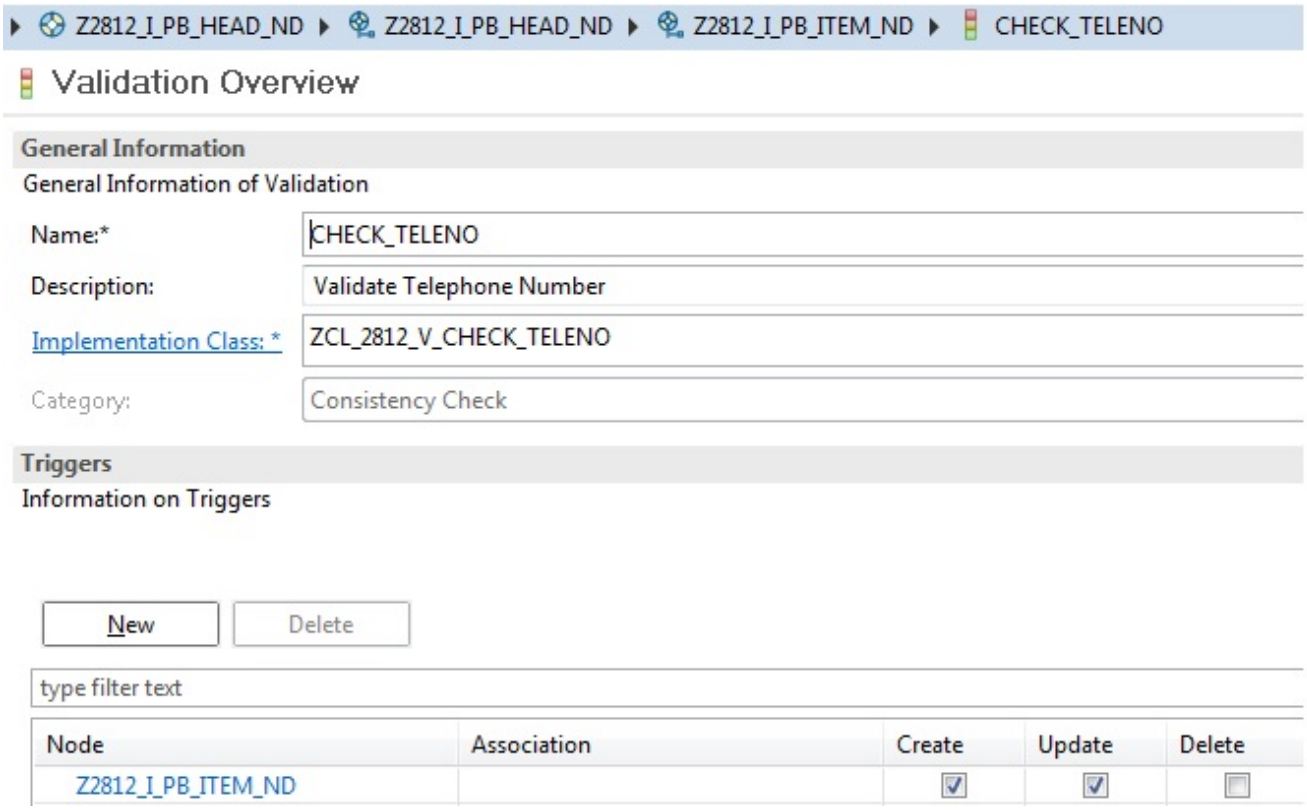
CHECK_TELENO

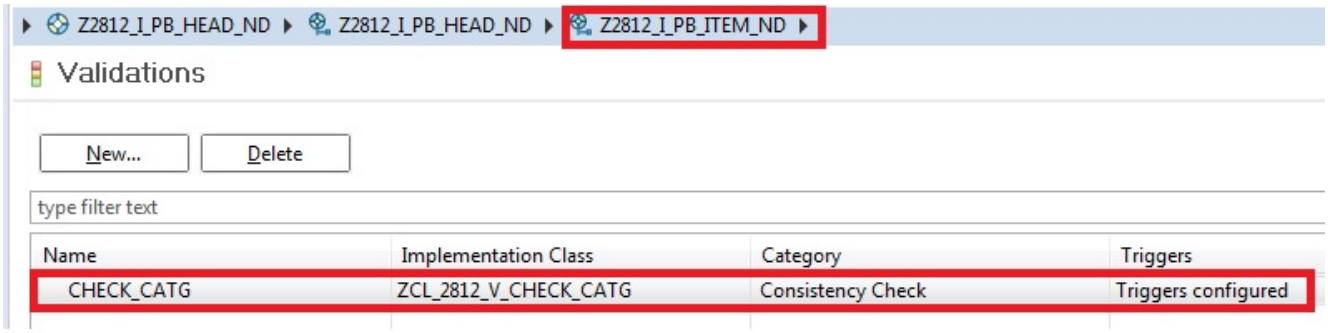
This will validate the the telephone number

Step 1: Navigate to *Business Object >> Go To Root Node >>Select Item Node*.

Step 2: *Select Validations Tab at the bottom or Click on Validations Link on Side*

Using remaining steps from CHECK_EMAIL validation and create validation **CHECK_TELENO**, it is associated with **Header Node**, consistency check and it is enabled for the action **Create** and **Update**.





Business Logic

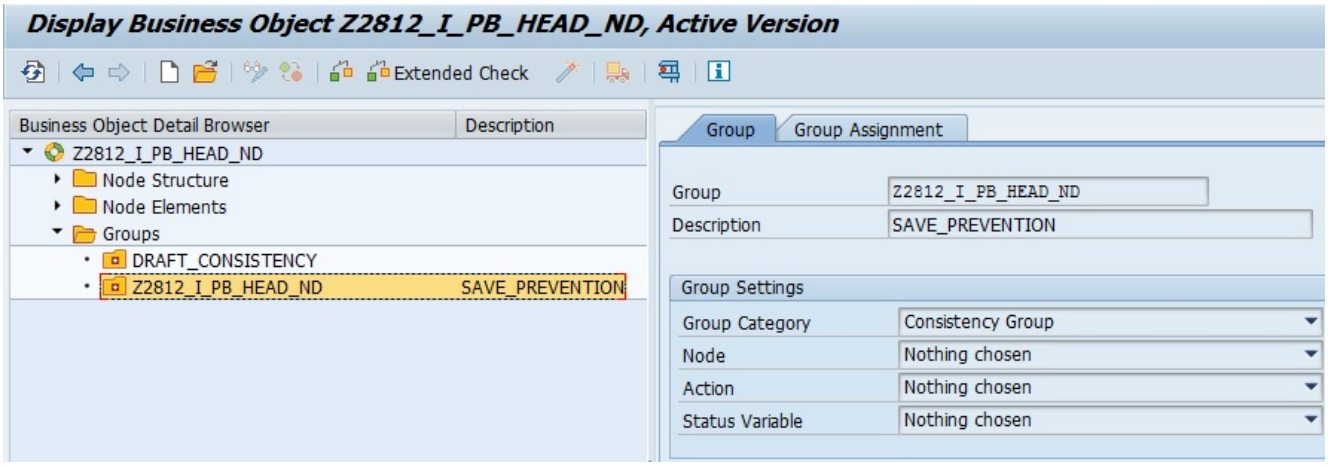
```
*---------------------------------------------------------------------*
* Method Name                  : /BOBF/IF_FRW_VALIDATION~EXECUTE
* Method description           : Validate Telephone#
*---------------------------------------------------------------------*
  METHOD /bobf/if_frw_validation~execute.
    " Internal tab for Item Data
    " Created using reference to Generated Table Type
    DATA(lt_item) = VALUE zt2812_i_pb_item_nd( ).
```

```
    " Get Phone Book Head Data
    io_read->retrieve(
      EXPORTING
        iv_node                = is_ctx-node_key " Node Name
        it_key                 = it_key          " Key Table
      IMPORTING
         et_data                   = lt_item         " Data
Return Structure
    ).

    " Instantiate Message Object
    IF eo_message IS NOT BOUND.
      eo_message = /bobf/cl_frw_factory=>get_message( ).
    ENDIF.

    LOOP AT lt_item REFERENCE INTO DATA(lr_item).
      " Validate
      me->validate_teleno(
        EXPORTING
          iv_telephone = lr_item->pb_telephone  " Telephone#
        IMPORTING
          es_msg     = DATA(ls_msg)             " Structure of
message
      ).

      " Add Message
      IF NOT ls_msg IS INITIAL.
        eo_message->add_message(
          EXPORTING
            is_msg       = ls_msg            " Message
            iv_node      = is_ctx-node_key   " Node
            iv_key       = lr_item->key      " Instance
        ).
      ENDIF.
    ENDLOOP.
  ENDMETHOD.
```

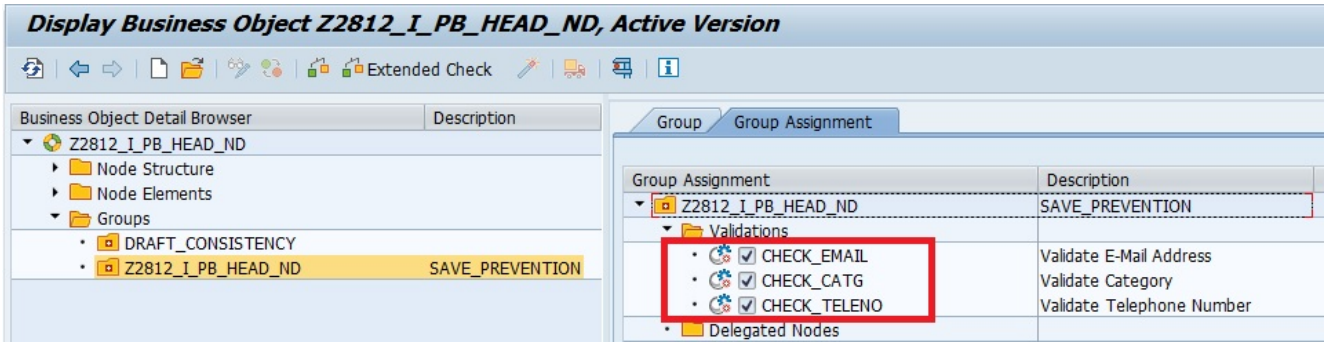
Validation has to be associated with the **consistency group**. A consistency group is executed before save and prevent it in case of errors, this way we do not need the same validations

as ASction Validation again.

In ADT based BOPF development, consistency group is auto created as soon as validation is activated. For this scenario, **Z2812_I_PB_HEAD_ND** (SAVE_PREVENTION) is generated.



Validations from both nodes are assigned to same consistency group.



[<< Development List]

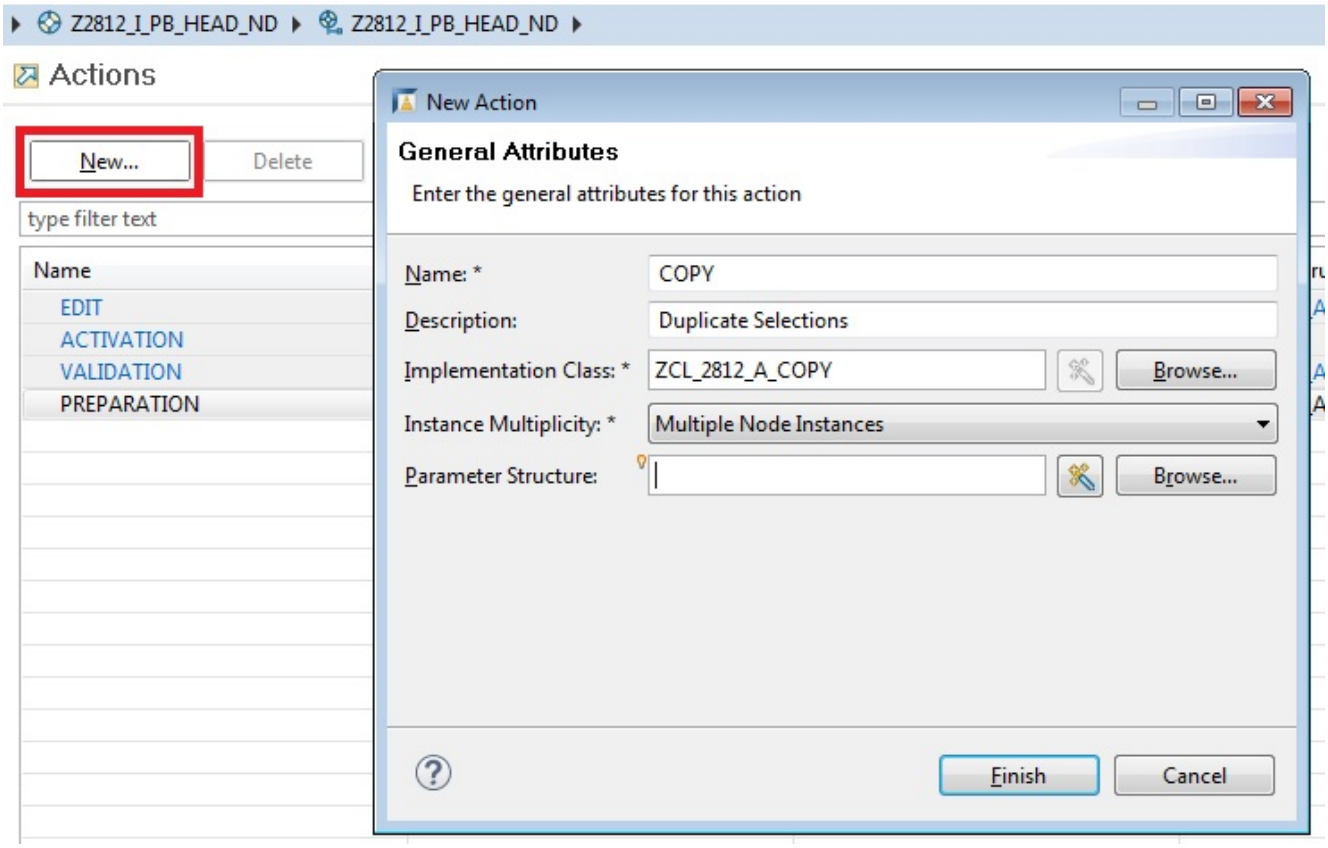
Actions

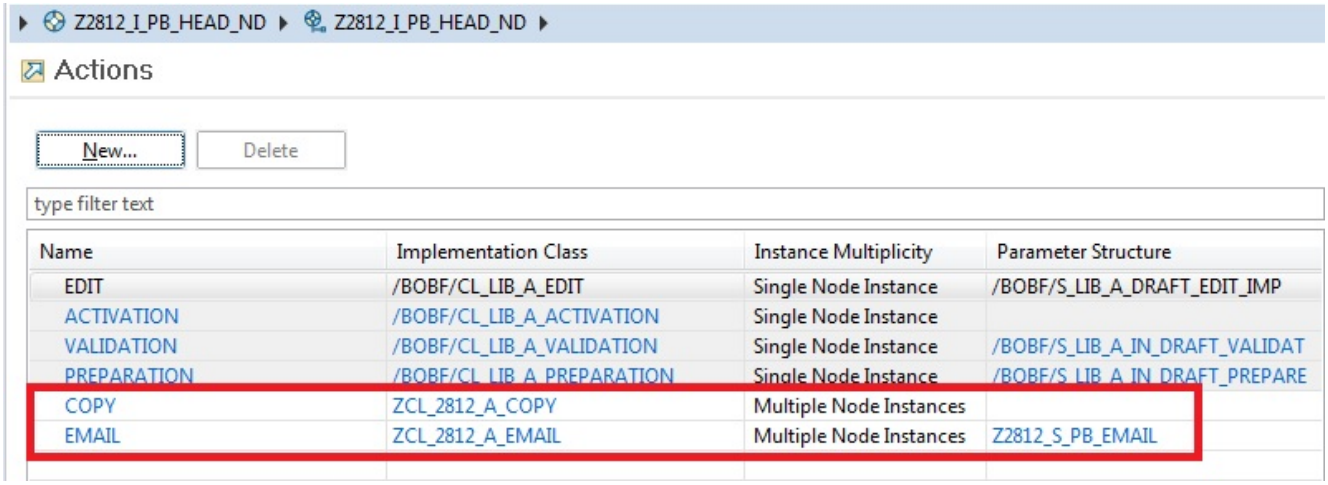
COPY

This will copy the selected phone book contacts.

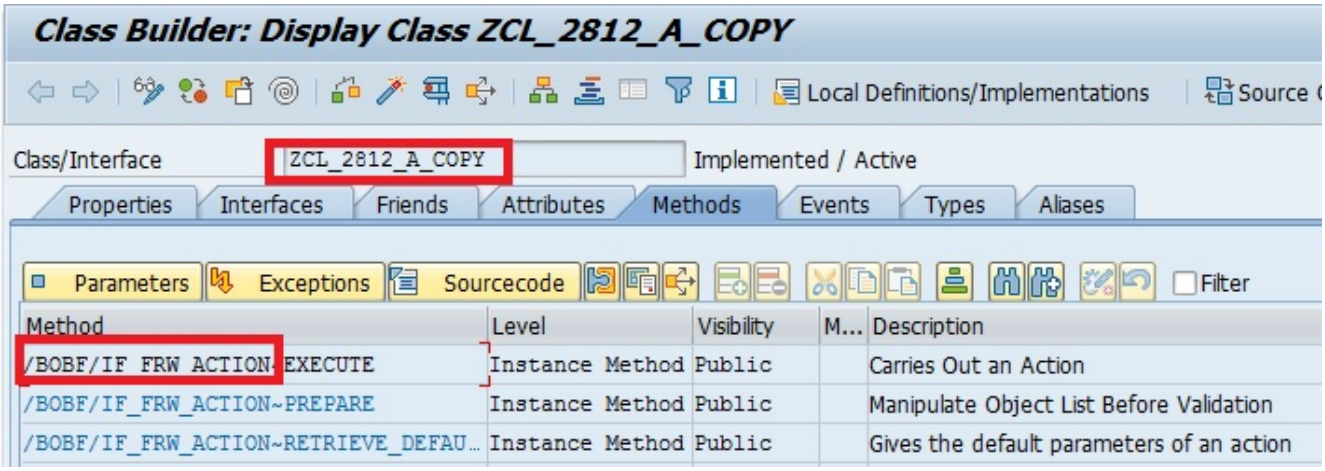
Step 1: Navigate to *Business Object >> Go To Root Node >> Select Actions Tab at the bottom or Click on Actions Link on Side*.

1. Once you are in Validations Tab then click on **New** to create an action.
2. Based on Name system will propose Implementation Class, but you can change the class name or re-generate the proposal.
3. Click on Finish
4. Save and Activate





Step 2: As soon as action is activated the associated implemented class is generated with the necessary interface, if it does not exists. Navigate to the class.



Step 3: Implement the business logic in the method **Execute**.

```
*---------------------------------------------------------------------*
* Method Name                : /BOBF/IF_FRW_ACTION~EXECUTE
* Method description         : Copy Order request
*---------------------------------------------------------------------*
  METHOD /bobf/if_frw_action~execute.
    DATA: lr_head_copy TYPE REF TO data,
          lr_item_copy TYPE REF TO data.

    " Internal tab for Header & Item Data
    " Created using reference to Generated Table Type
    DATA(lt_head) = VALUE zt2812_i_pb_head_nd( ).
    DATA(lt_item) = VALUE zt2812_i_pb_item_nd( ).

    " Get Phone Book Head Data
    io_read->retrieve(
      EXPORTING
        iv_node                 = is_ctx-node_key " Node Name
        it_key                  = it_key          " Key Table
      IMPORTING
         et_data                    = lt_head         " Data
Return Structure
    ).

    " Get Phone Book Item Data
```

```
    io_read->retrieve_by_association(
      EXPORTING
        iv_node                 = is_ctx-node_key  " Node Name
        it_key                  = it_key           " Key Table
                    iv_association                          =
zif_2812_i_pb_head_nd_c=>sc_association-z2812_i_pb_head_nd-
_itemdata " Name of Association
        iv_fill_data            = abap_true
      IMPORTING
         et_data                   = lt_item          " Data
Return Structure
    ).

    " For Each Node Instance
    LOOP AT lt_head REFERENCE INTO DATA(lr_head).
      ASSIGN lr_head->* TO FIELD-SYMBOL(<fs_head>).

      " Create Copy of Head
      DATA(ls_head_copy) = VALUE zs2812_i_pb_head_nd( ).
      ls_head_copy-pb_email_id   = <fs_head>-pb_email_id.
      ls_head_copy-pb_first_name = <fs_head>-pb_first_name.
      ls_head_copy-pb_last_name  = <fs_head>-pb_last_name.


      " Create Data Ref to Copy of Head
      CREATE DATA lr_head_copy TYPE zs2812_i_pb_head_nd.
      ASSIGN lr_head_copy->* TO FIELD-SYMBOL(<fs_head_copy>).
      IF <fs_head_copy> IS ASSIGNED.
        <fs_head_copy> = ls_head_copy.
      ENDIF.

      " Create New Phone Book Entry
      io_modify->create(
        EXPORTING
          iv_node            =  is_ctx-node_key             "
Node to Create
          is_data            =  lr_head_copy                "
Data
        IMPORTING
          ev_key             = DATA(lv_pb_copy_key)
      ).
```

```
       LOOP AT lt_item REFERENCE INTO DATA(lr_item) WHERE
parent_key = lr_head->key.
        ASSIGN lr_item->* TO FIELD-SYMBOL(<fs_item>).

        " Create Copy of Item
        DATA(ls_item_copy) = VALUE zs2812_i_pb_item_nd( ).
        ls_item_copy-pb_category  = <fs_item>-pb_category.
        ls_item_copy-pb_telephone = <fs_item>-pb_telephone.

        " Create Reference to Item Copy
        CREATE DATA lr_item_copy TYPE zs2812_i_pb_item_nd.
                   ASSIGN  lr_item_copy->*  TO  FIELD-
SYMBOL(<fs_item_copy>).
        IF <fs_item_copy> IS ASSIGNED.
          <fs_item_copy> = ls_item_copy.
        ENDIF.

        io_modify->create(
          EXPORTING
                          iv_node                         =
zif_2812_i_pb_head_nd_c=>sc_node-z2812_i_pb_item_nd
" Node to Create
                 is_data                =  lr_item_copy
" Data
                              iv_assoc_key                =
zif_2812_i_pb_head_nd_c=>sc_association-z2812_i_pb_head_nd-
_itemdata " Association
                                    iv_source_node_key  =
zif_2812_i_pb_head_nd_c=>sc_node-z2812_i_pb_head_nd
" Parent Node
                iv_source_key        =  lv_pb_copy_key
" NodeID of Parent Instance
        ).
      ENDLOOP.
    ENDLOOP.
  ENDMETHOD.
```

[<< Development List]

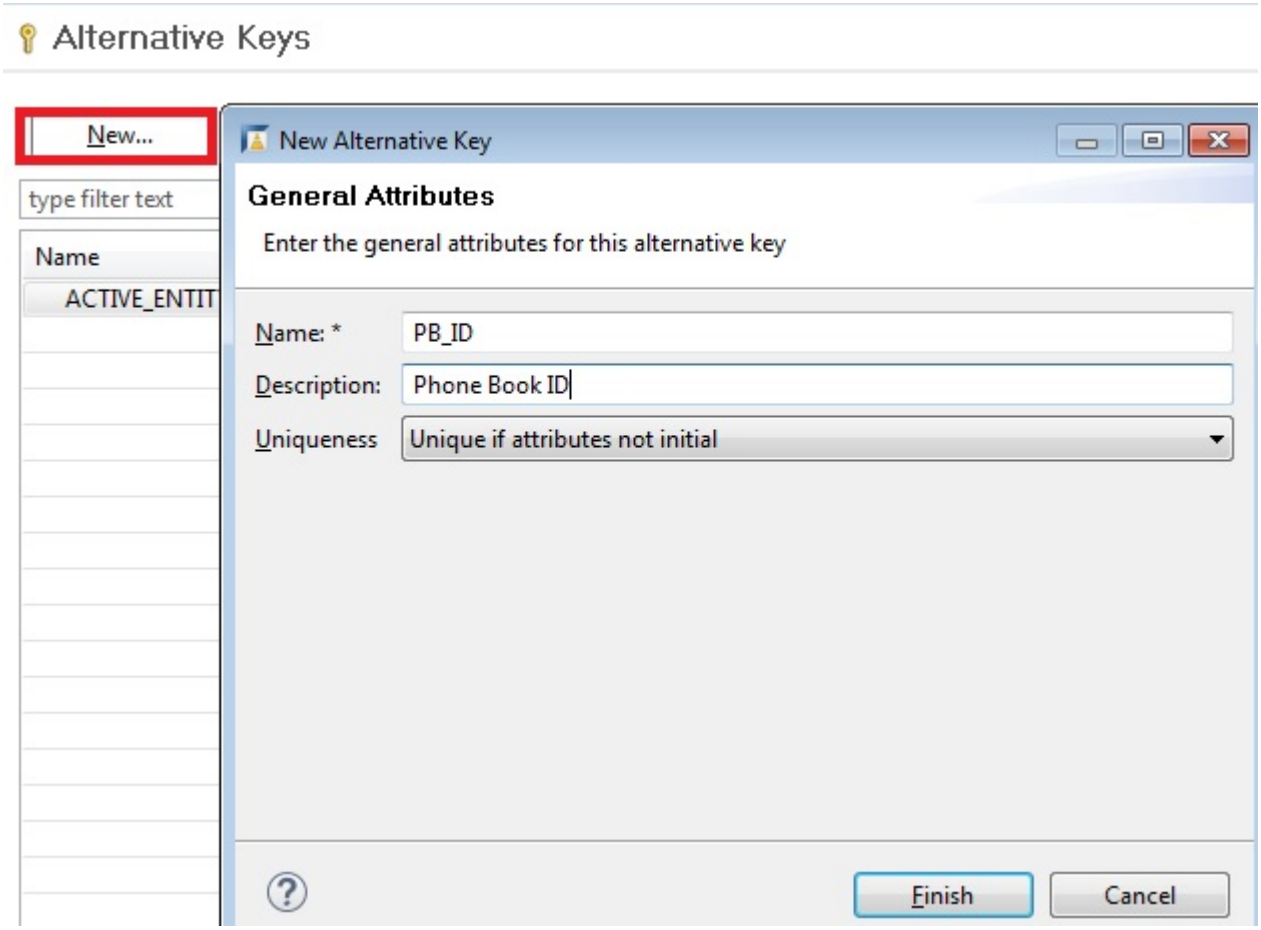
Alternative keys

PB_ID

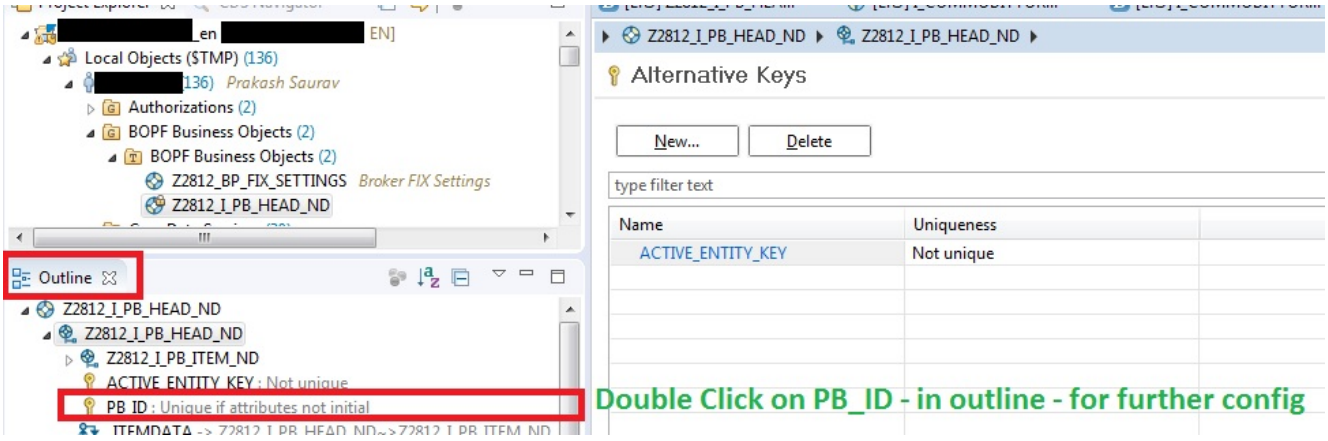
This key will allow us to find node instance using Phone Book ID.

Step 1: Navigate to *Business Object >> Go To Root Node >> Select*
Alternative Keys Tab at the bottom or Click on Alternative Keys Link on Side.

1. Once you are in Alternative Keys Tab then click on **New** to create.
2. Click on Finish

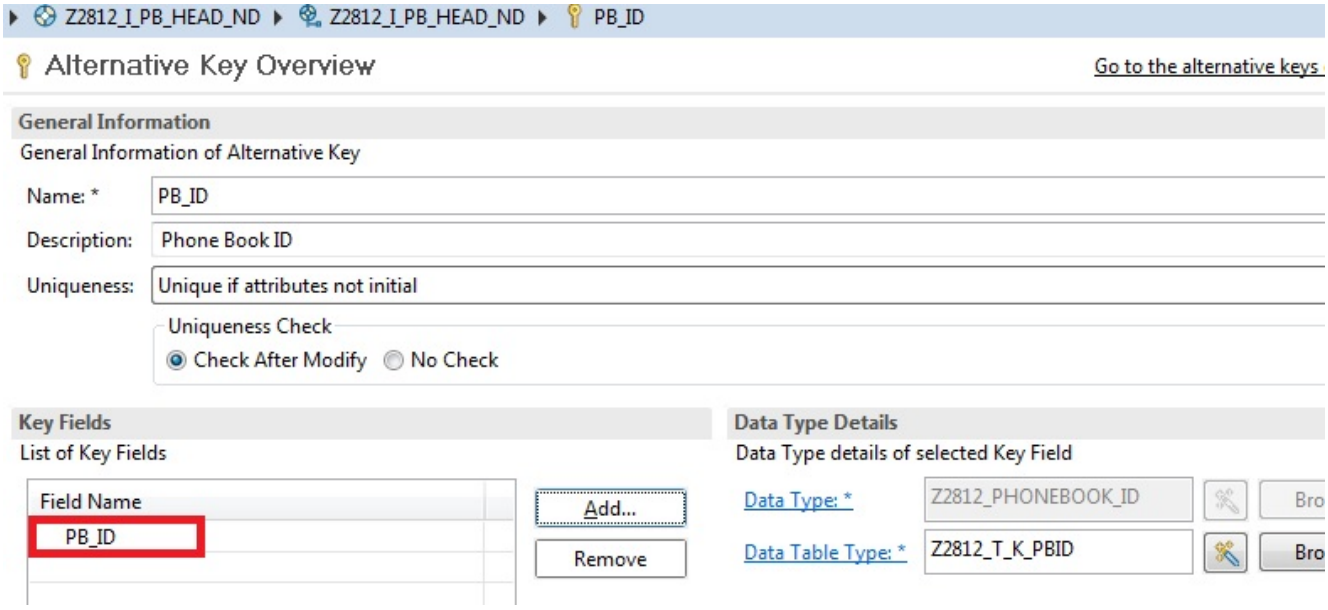


Step 2: You can double click on the new Alternative Key from **Outline View** or you can select the new alternative key from **Node**, to further configure it.



Step 3:

1. Maintain Field Name (from Node Attributes)
2. Save and Activate



[<< Development List]

[<< Top]

Test BOPF object using BOBT

---

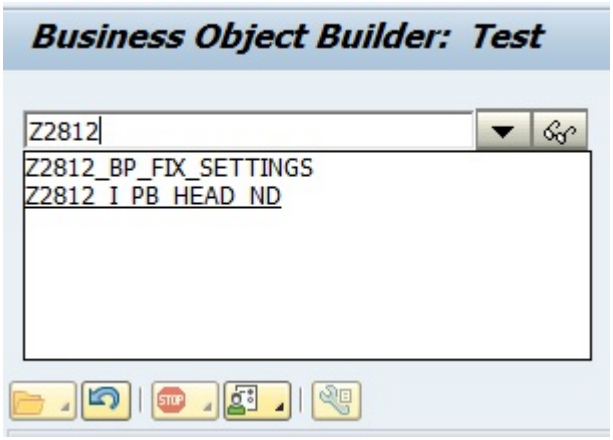
1. Design / Prototype
2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)

5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. **Test BOPF Object using BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

This blog showcase Business object testing using transaction **BOBT**. Step number and count mentioned is very specific (not a generic sequence) to the application scenario. It is just to show, how to carry out the different activities in BOBT.

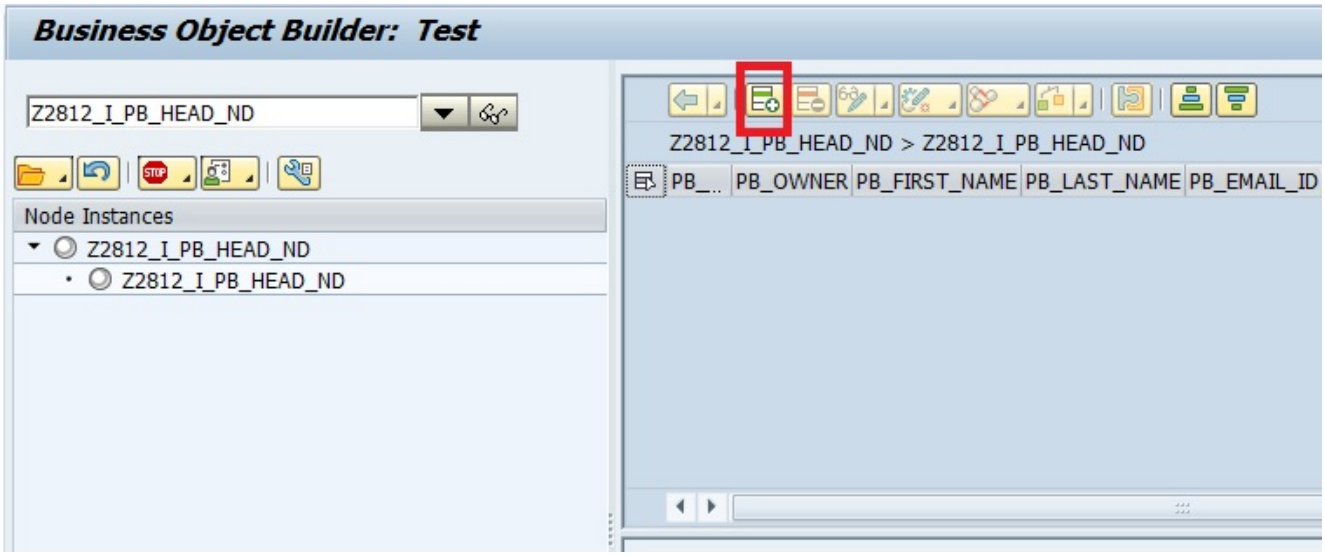
Step 1: Go to Transaction '**BOBT**

Step 2: Enter the Business Object name as shown below

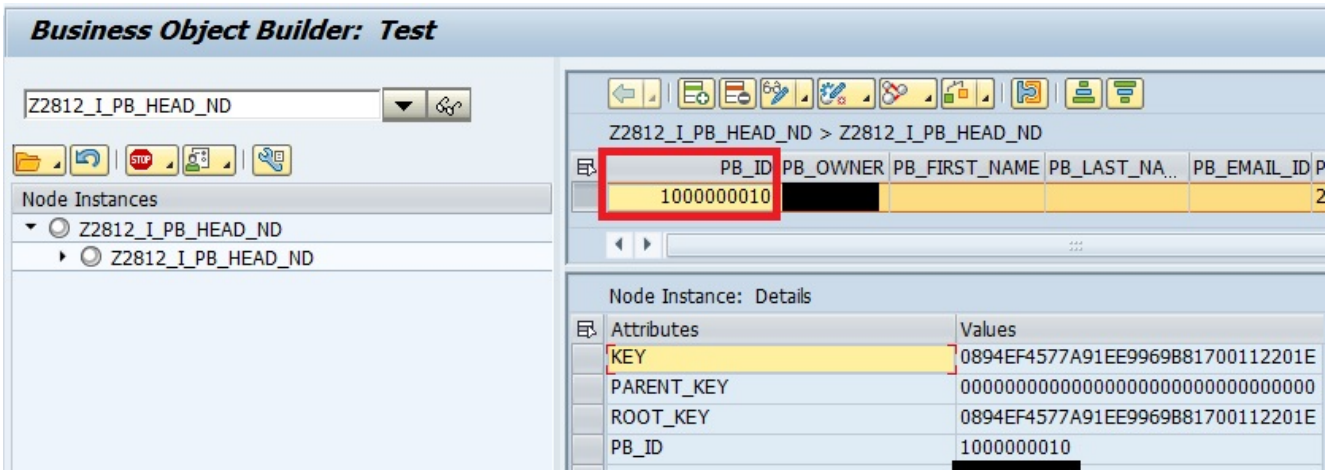


Step 3: Click on 'New Entry' to create new instance of the

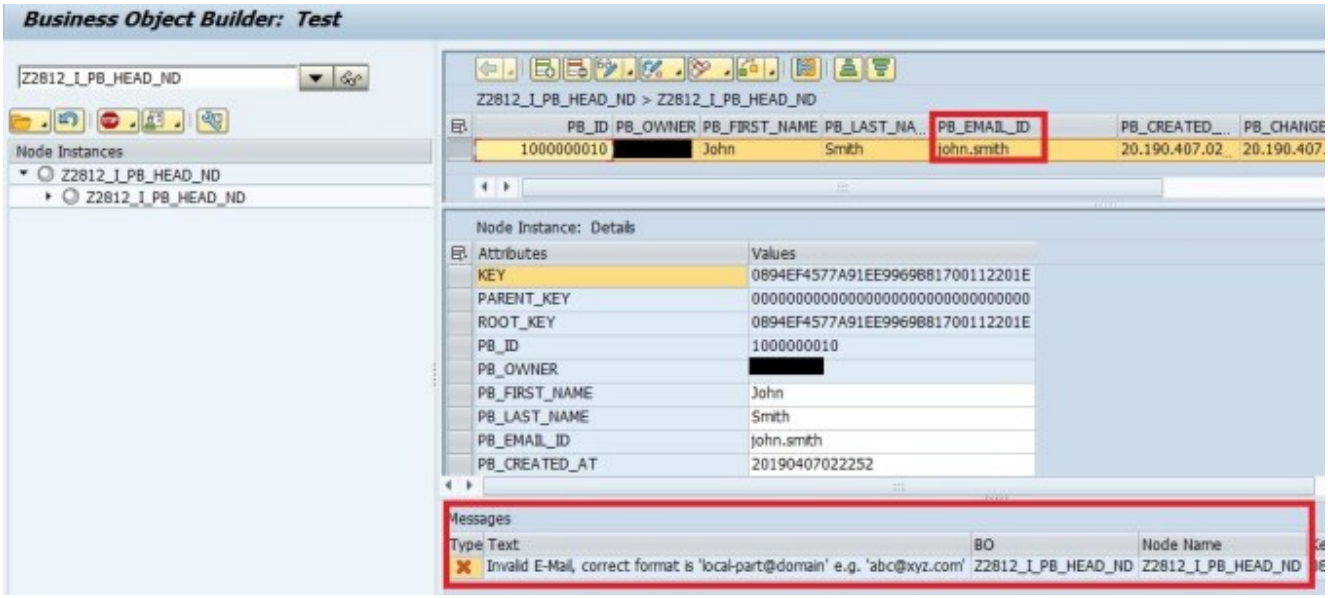
node.



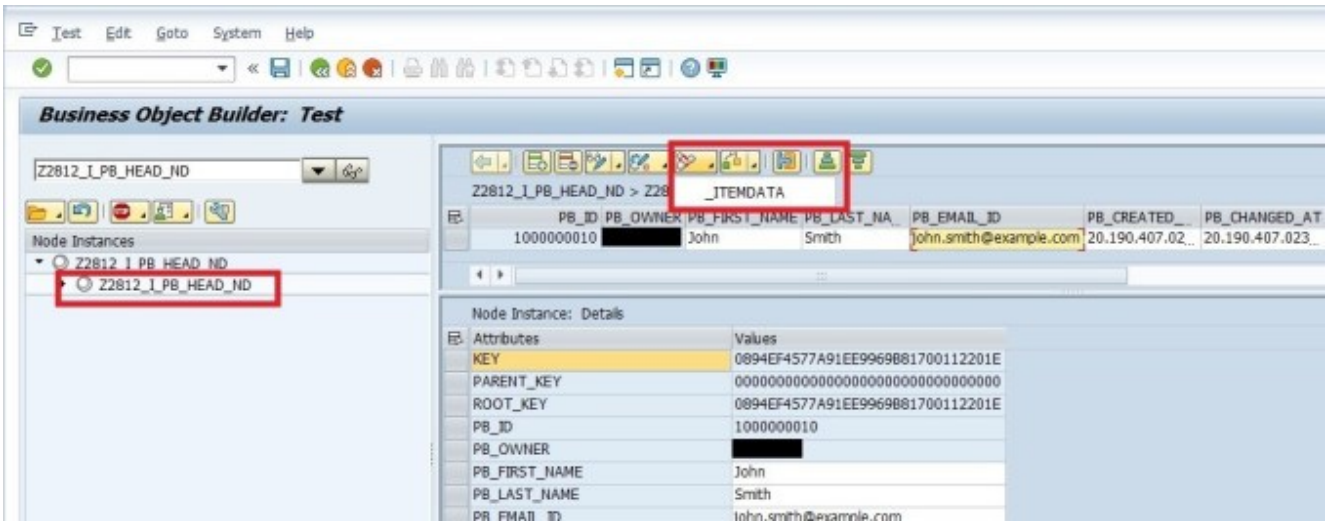
As soon as a new node instance is created the *determination* to get Phone Book ID is executed and the same is assigned and updated along with admin data.



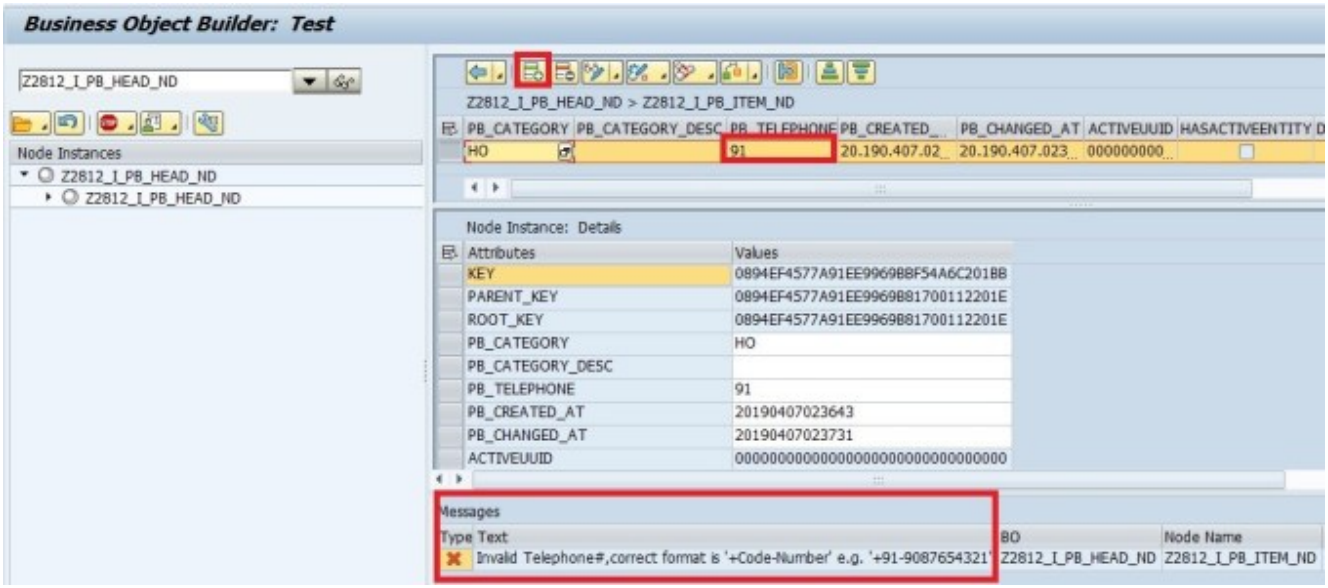
Step 4: If you have header level validation the enter wrong value to trigger the error message. In this scenario, invalid E-Mail ID is entered and the validation check fails.



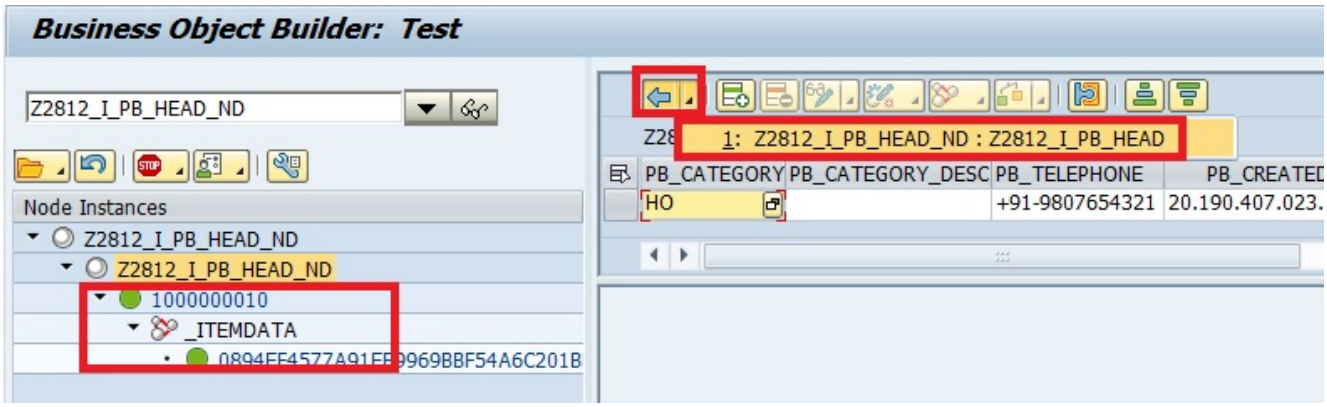
Step 5: Go to Item Node. You can navigate to an item using the association or by instance tree in the left pane.



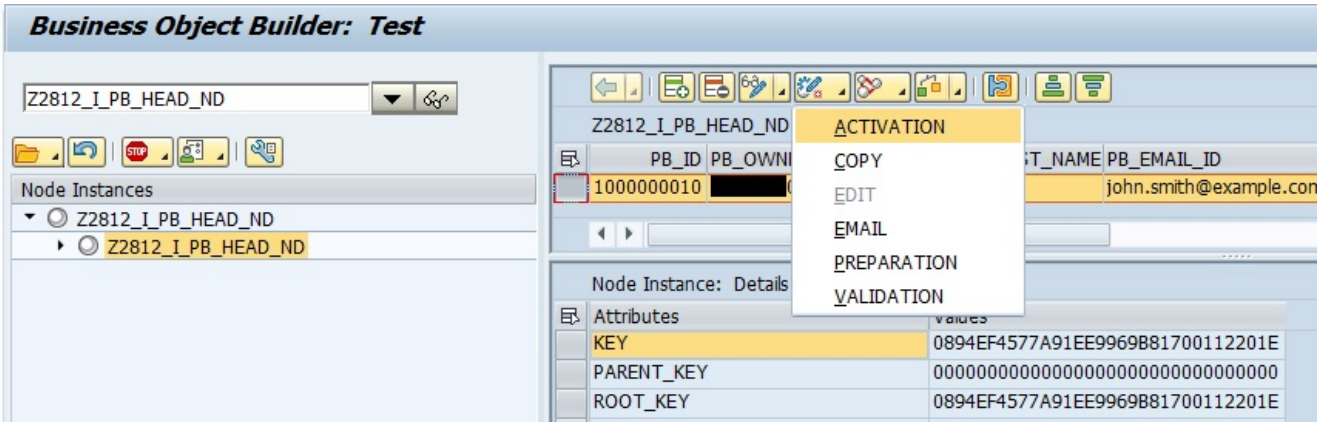
Step 6: Create item node instance and If you have item level validation the enter wrong value to trigger the error message. In this scenario, the invalid telephone number is entered and the validation check fails.



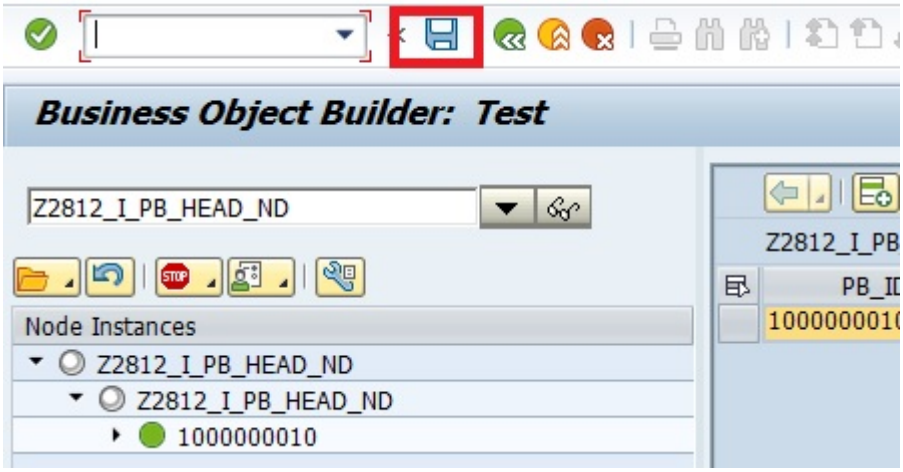
Step 7: Navigate back to Parent Node, using Back or Association or Node Tree.



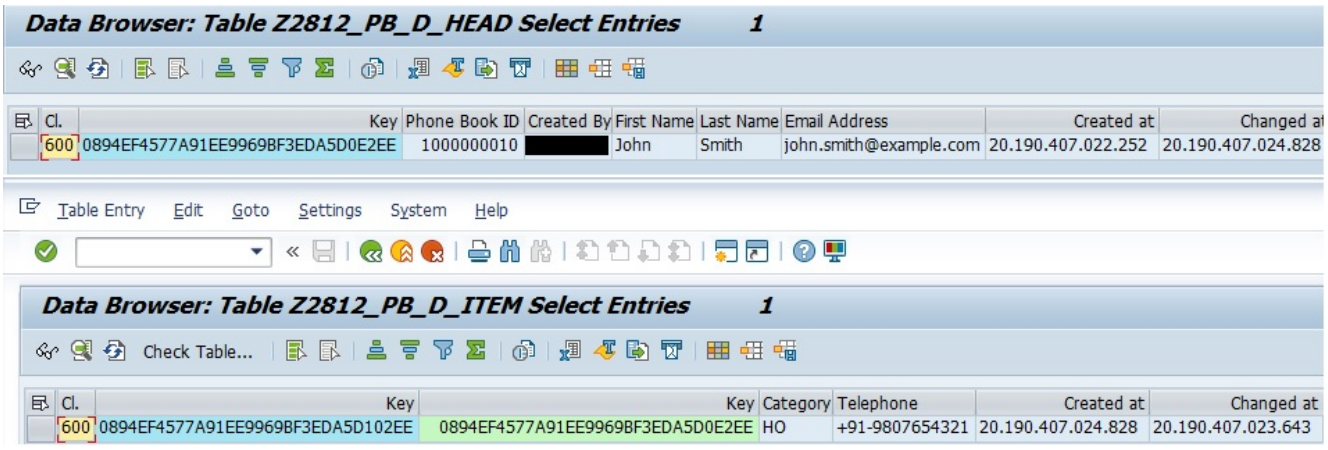
Step 8: Select the node and Activate it. It will prepare the node for persistence.



Step 9: Save the transaction to persist the entry.

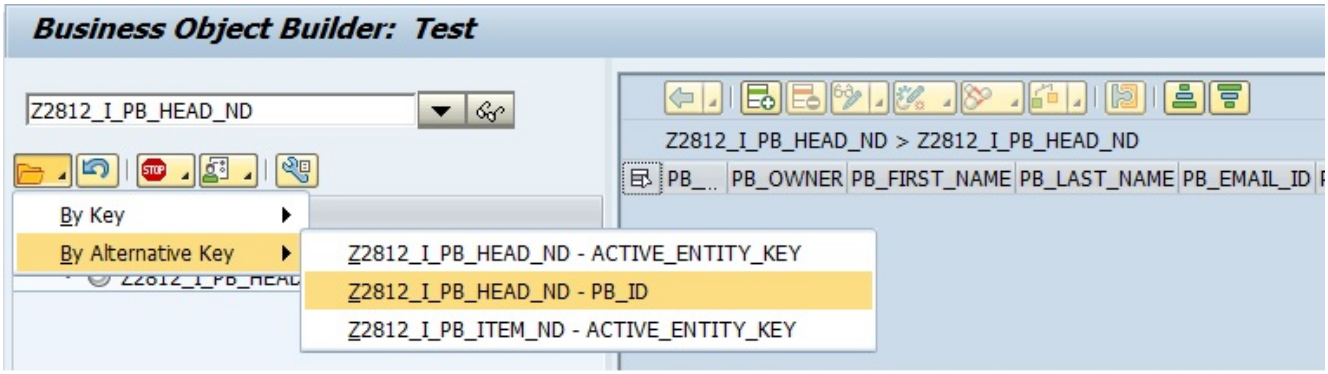


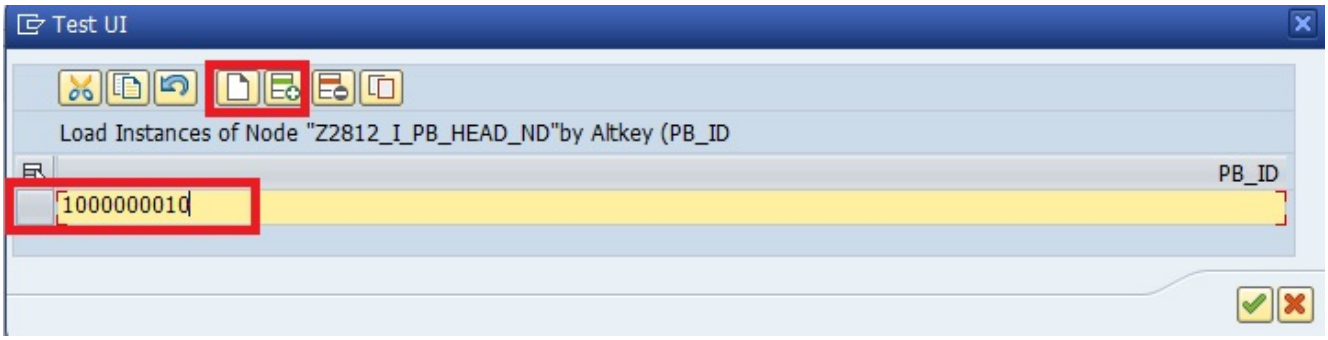
Step 10: Verify the entries persisted in database.



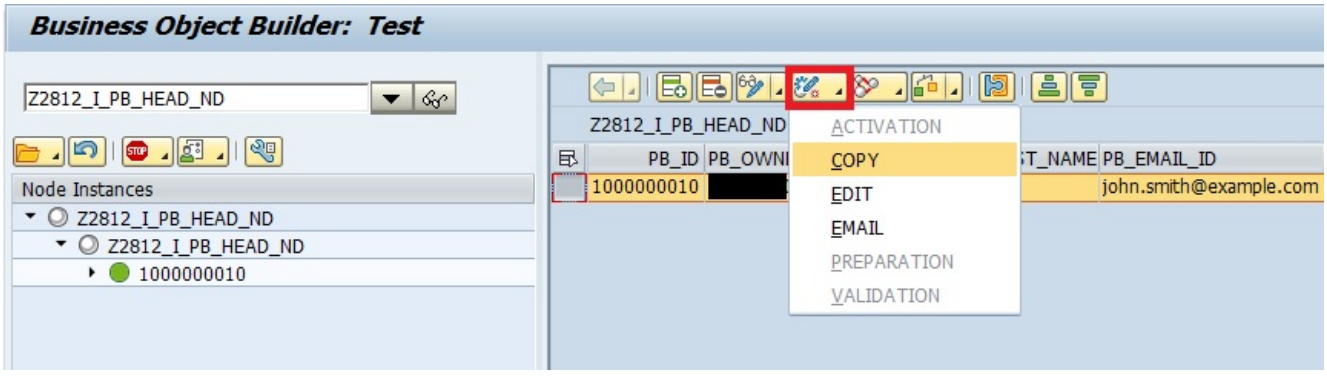
Step 11: Go to **BOBT** and open the Business Object, then use the

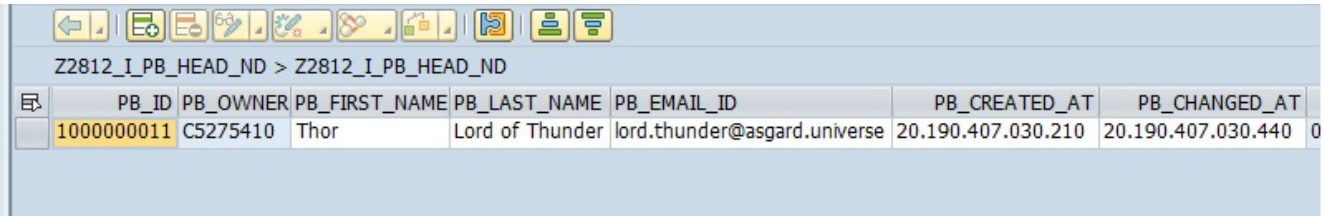
Alternative Key Phone Book ID to get the node instance.





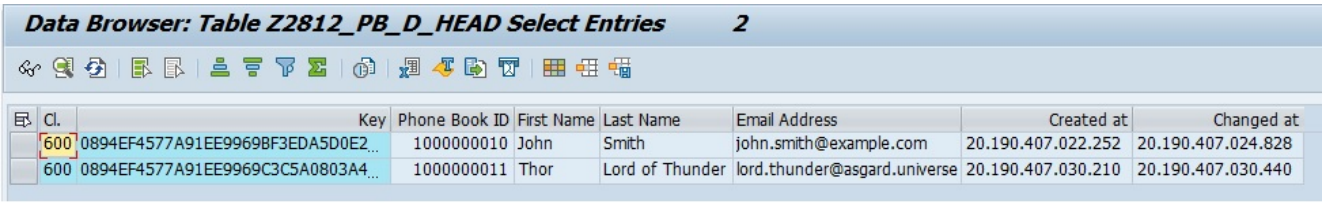
Step 12: If you have any add-on action for the given business object then execute the same. In this scenario, node instance is selected and **COPY** is executed. Activate and save the data, then check the new entry in database.





Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND

PB_ID	PB_OWNER	PB_FIRST_NAME	PB_LAST_NAME	PB_EMAIL_ID	PB_CREATED_AT	PB_CHANGED_AT
1000000011	C5275410	Thor	Lord of Thunder	lord.thunder@asgard.universe	20.190.407.030.210	20.190.407.030.440



Data Browser: Table Z2812_PB_D_HEAD Select Entries 2

Cl.	Key	Phone Book ID	First Name	Last Name	Email Address	Created at	Changed at
600	0894EF4577A91EE9969BF3EDA5D0E2...	1000000010	John	Smith	john.smith@example.com	20.190.407.022.252	20.190.407.024.828
600	0894EF4577A91EE9969C3C5A0803A4...	1000000011	Thor	Lord of Thunder	lord.thunder@asgard.universe	20.190.407.030.210	20.190.407.030.440

[<< Top](#)

Access Control (CDS)

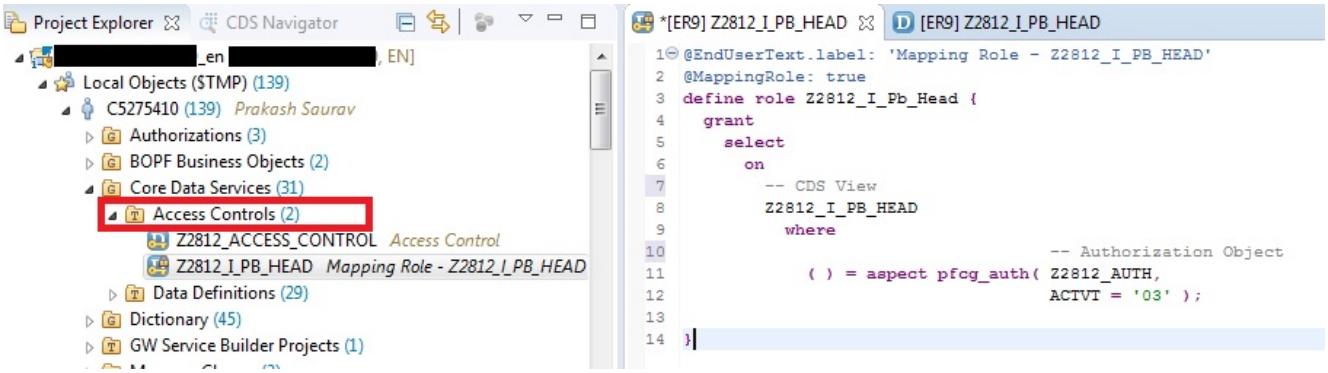
---

10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. UI development

DCL Source

With the help of Core Data Services, we can also define and generate DCL (Data Control Language) Source. It is used to restrict access to data.

Create an Access Control object '**Z2812_I_PB_HEAD**' (I practice to create the access control with the same name as the DDL Source. It helps to easily identify the CDS View used inside the access control).




```
@EndUserText.label: 'Mapping Role - Z2812_I_PB_HEAD'
@MappingRole: true
define role Z2812_I_Pb_Head {
  grant
    select
      on
        -- CDS View
        Z2812_I_PB_HEAD
          where
                                    -- Authorization Object
```

```
            ( ) = aspect pfcg_auth( Z2812_AUTH,
                                    ACTVT = '03' );
}
```

No CDS element is specified in the above example. CDS access control prevents data from being read in full if the current user does not have at least an authorization for the authorization object with the activity “03”. You can also restrict data with control over specific fields, refer below link for more details.
Read More about PFCG_AUTH

The below control statement used in CDS View triggers the check using using access control.

```
@AccessControl.authorizationCheck: #CHECK
```

Similarly, create access control for all the required views with the required authorization.

[<< Top]

User Interface Development

---

1. Design / Prototype

2. Create Dictionary Objects
3. Create Interface View (CDS)
4. Business Object Generation (CDS – Object Model Annotations)
5. Generated Business Object
6. BOPF – Development
    - Determination
    - Validation
    - Action
    - Alternative Key
7. Test BOPF Object using **BOBT**
8. Access Control (CDS)
9. Consumption View (CDS)
10. OData Service Generation and Registration
11. Test OData Service using SAP Gateway Client
12. **UI development**

Frontend Side consumption of BOPF

- FBI (FPM BOPF Integration)
- Fiori Elements

FBI (FPM BOPF Integration)

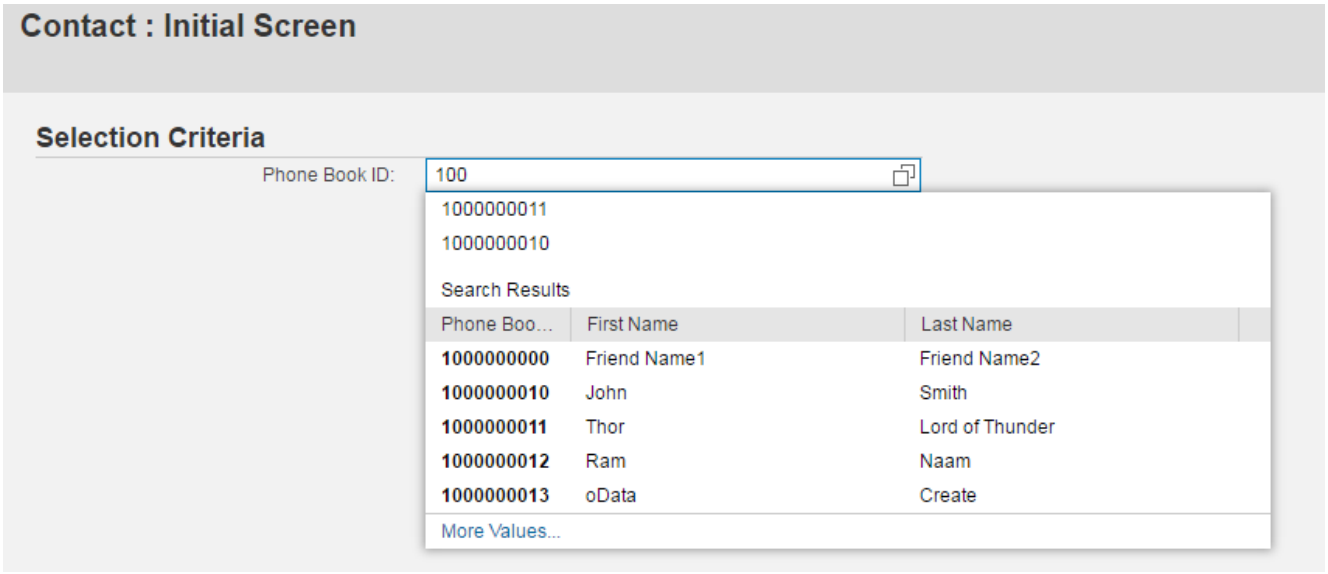
I have followed each every single steps mention in the below blog to develop the integration. It is really helpful, please follow the same. Floorplan used for the application is OVP

(Overview Page Floorplan).
Getting Started with FPM and BOPF Integration

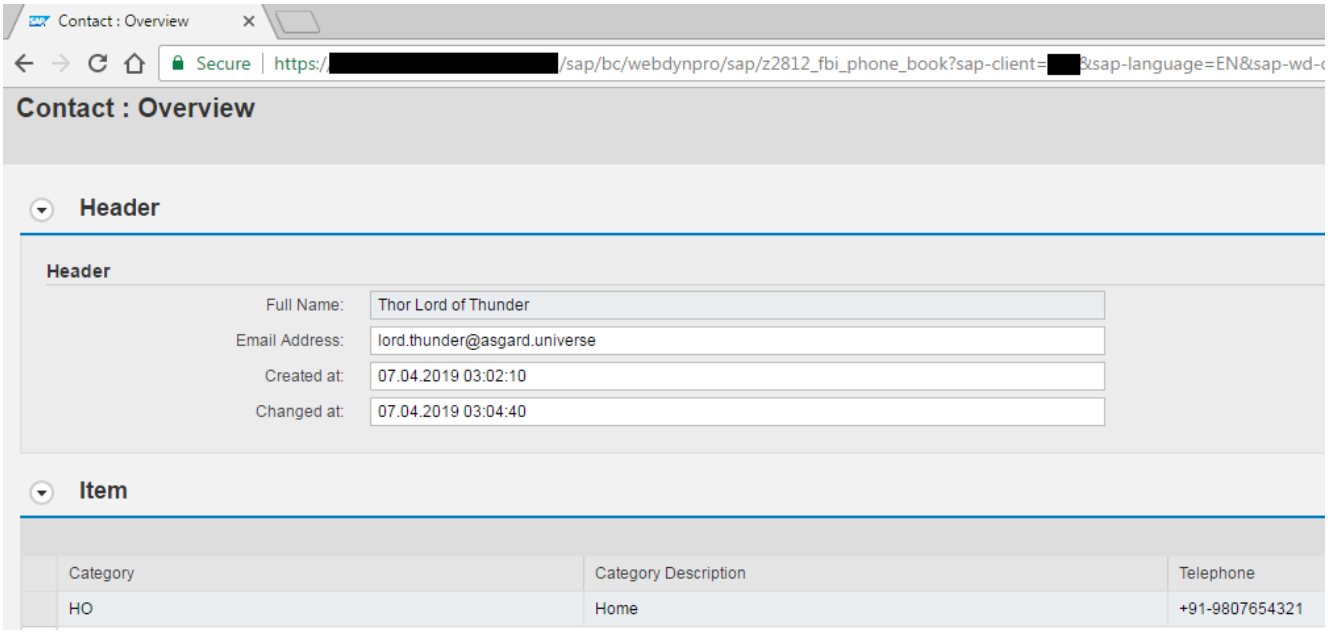
Application (Display Only)

<< Frontend Consumption

Initial Screen



Dynamic Search & History Enabled



Overview Page

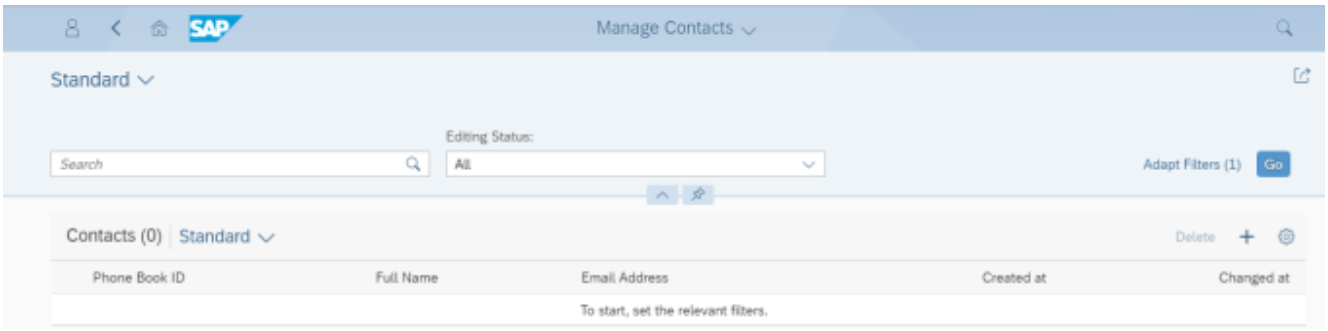
Fiori Elements

List Report application type was used to create the application. Refer the video in below link, it is really simple and easy.

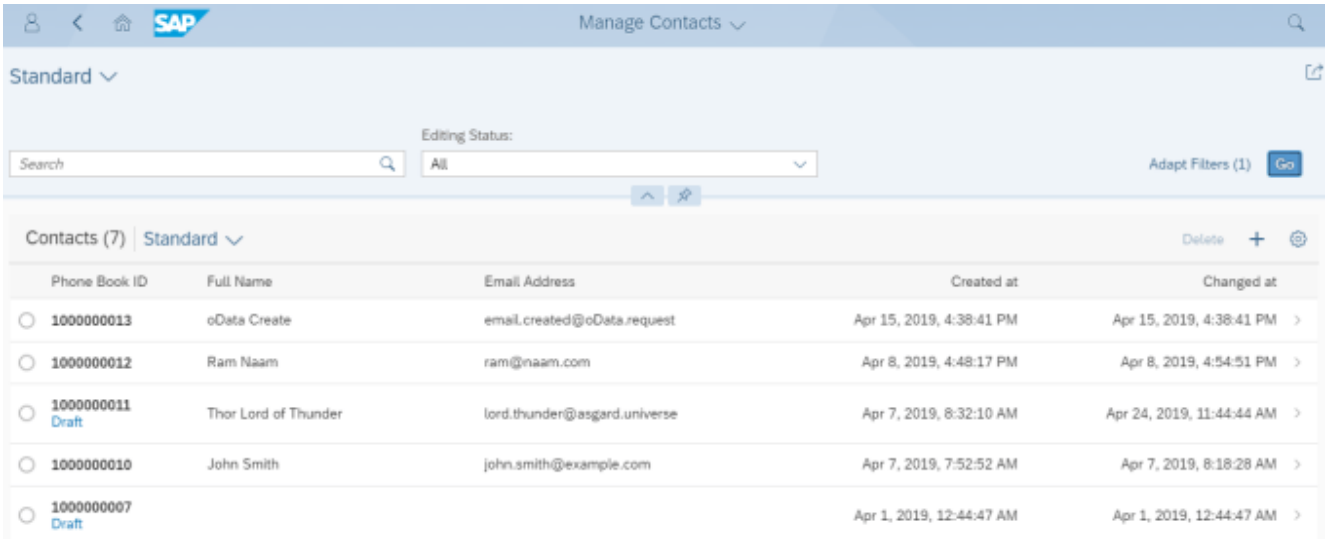
Creating a List Report with Fiori Elements

For more insight, please do refer to SPAUI5 Development and Demo Kit.
[Developing Apps with SAP Fiori Elements]

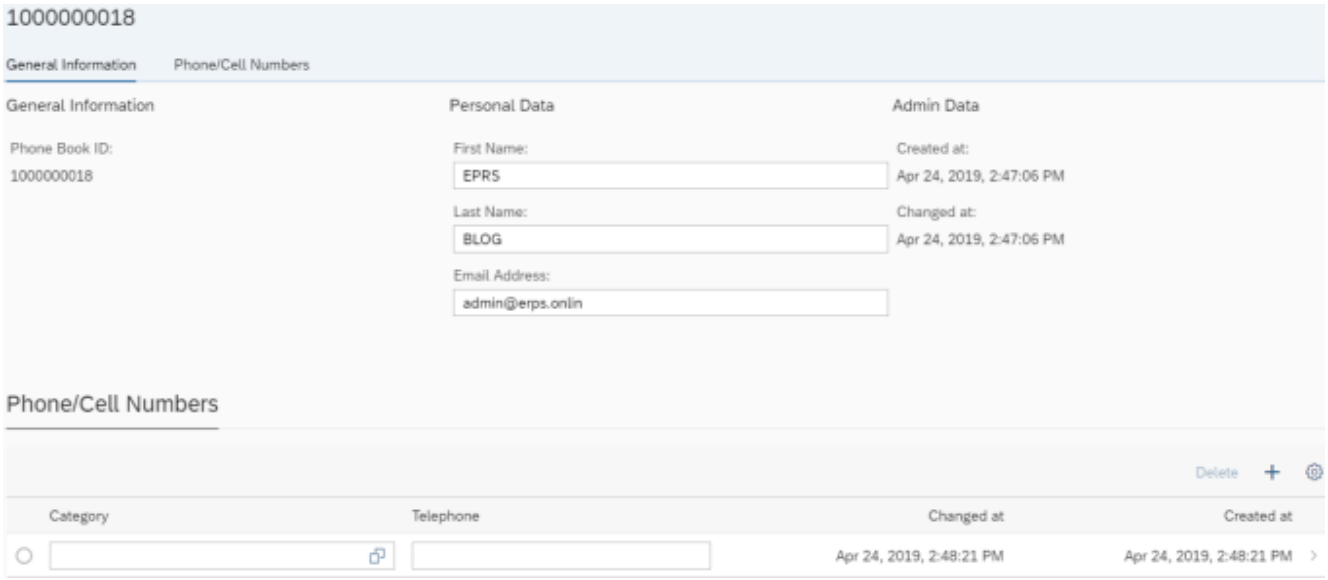
Application (CRUD + Draft Enabled)



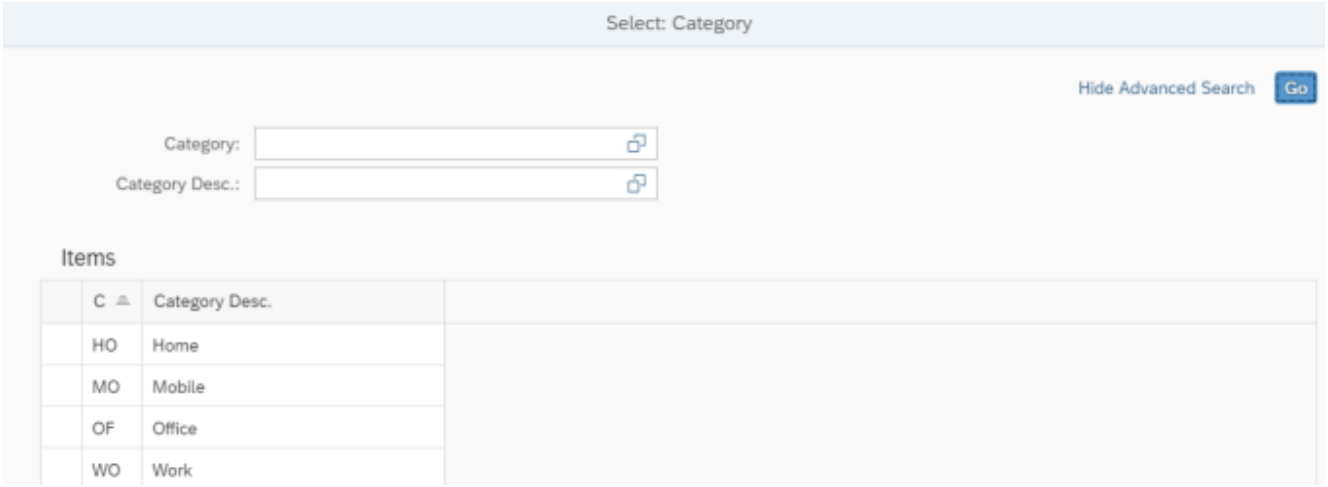
Initial List Screen (Click on Go to Load Data)



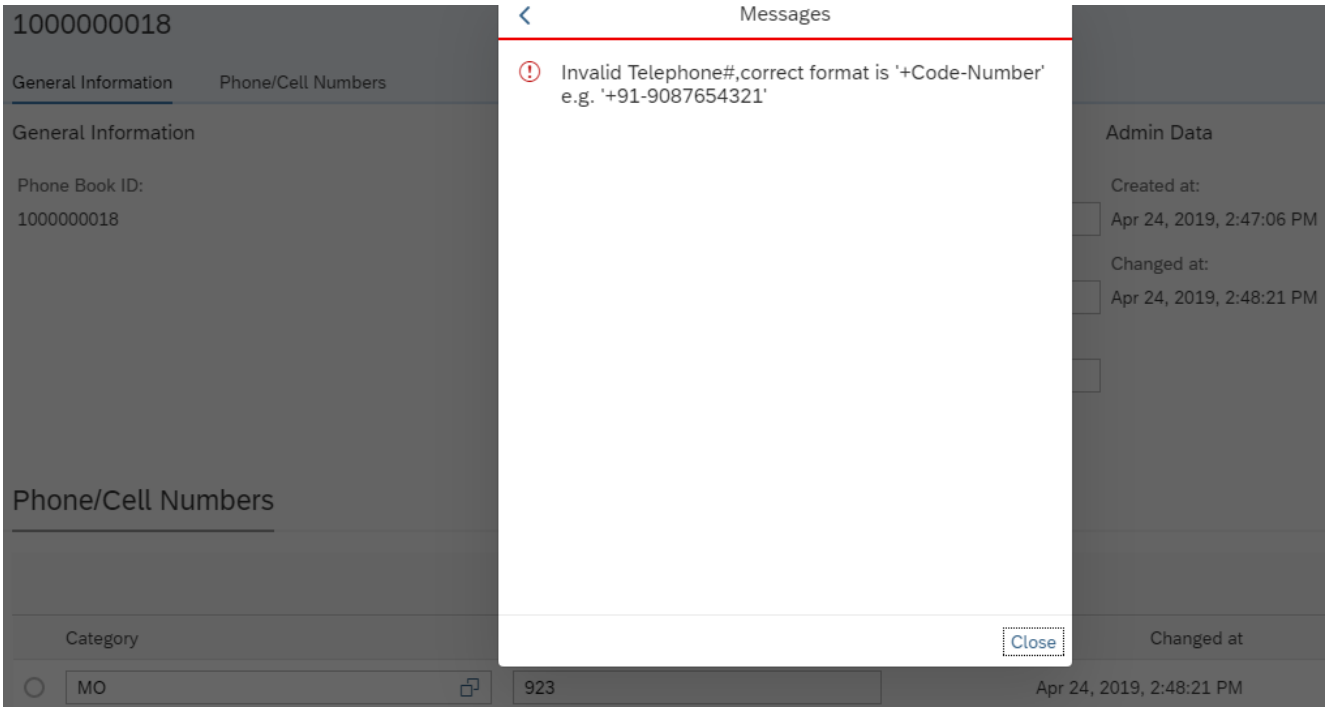
List Report (Draft Enabled) (Click on ‘+’ to Create) (On Item Selection ‘Delete’ is enabled)



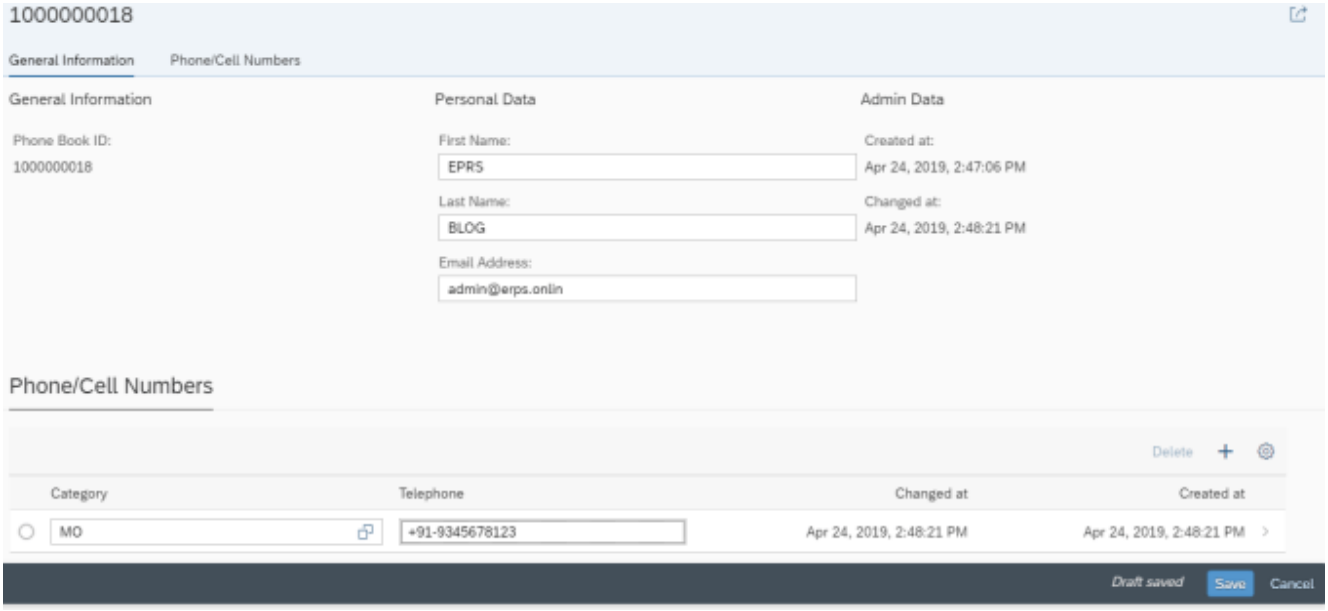
New Phone Book ID is assigned and data can be filled



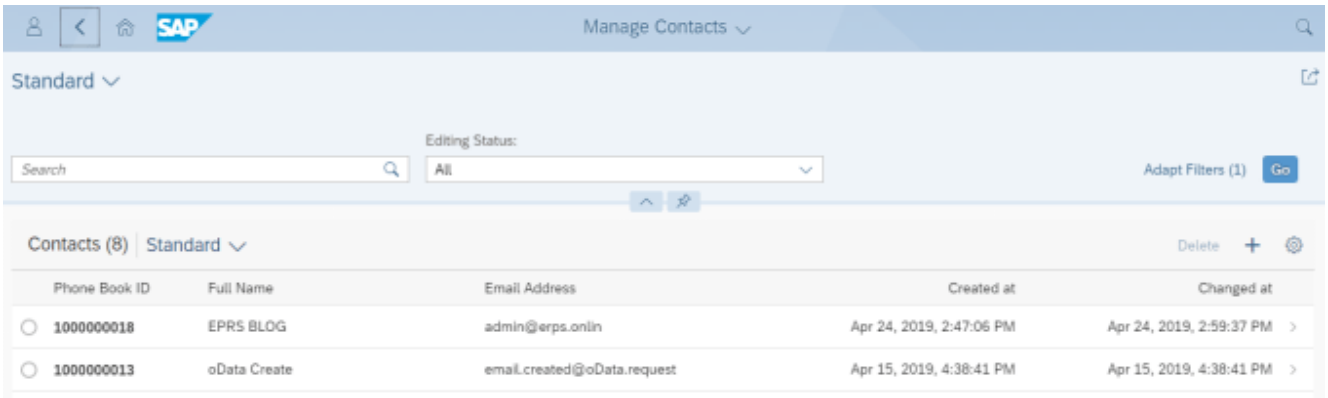
Value Help Supplied for Category



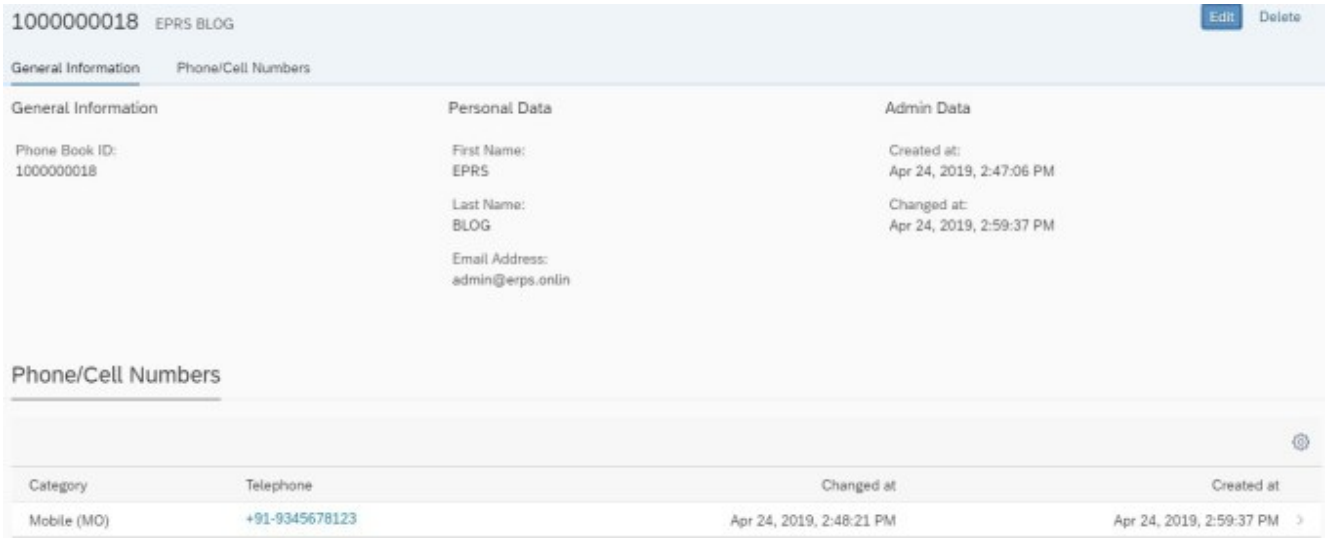
Validations are Working



Save



New Contact is Saved



Navigate to Contact Item (Edit and Delete Options are Visible)

[<< Frontend Consumption]

[<< Top]