

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

Phone Book ID	Full Name	Email
100000011	Social Sharma	Social.Sharma@build.me

Desktop – Object Page

The screenshot shows a contact record for 'Social Sharma'. At the top right are 'Edit', 'Delete', and a share icon. Below the name, it says 'First Name: Social' and 'E-Mail: Social.Sharma@build.me'. Under 'Last Name', it shows 'Social SHarma'. There are two tabs: 'GENERAL' (underlined) and 'NUMBERS'. The 'GENERAL' tab contains 'Admin Data' with creation and last change dates. The 'NUMBERS' tab lists one phone number: '+91-9087654321' under 'Category: Home'.

Category	Phone Number	Created On	Last Changed On
Home	+91-9087654321	April 16, 2019 at 9:32:50 AM GMT+05:30	April 16, 2019 at 9:35:50 AM GMT+05:30

Mobile – List Report

Default ▾

^ ↗

Contacts (1) Delete + ⚙

Full Name

Social Sharma >

Phone Book ID:
1000000011

E-Mail:
Social.Sharma@build.me

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	<input checked="" type="checkbox"/>	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
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT
PB_HEAD_UUID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/BOBF/UUID	RAW
PB_ID	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_PHONEBOOK...	NUMC
PB_OWNER	<input type="checkbox"/>	<input type="checkbox"/>	ERNAME	CHAR
PB_FIRST_NAME	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_FIRSTNAME	CHAR
PB_LAST_NAME	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_LASTNAME	CHAR
PB_EMAIL_ID	<input type="checkbox"/>	<input type="checkbox"/>	AD_SMTPADR	CHAR
PB_CREATED_AT	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_CREATED_AT	DEC
PB_CHANGED_AT	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_CHANGED_AT	DEC

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

Field	Key	Ini...	Data Element	Data Type	Length	Deci...	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0	Client
PB_ITEM_UUID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	/BOBF/UUID	RAW	16	0	UUID serving as key (parent key, root key)
PB_HEAD_UUID	<input type="checkbox"/>	<input type="checkbox"/>	/BOBF/UUID	RAW	16	0	UUID serving as key (parent key, root key)
PB_CATEGORY	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_PB_CATEG...	CHAR	2	0	Category of Contact Numbers
PB_TELEPHONE	<input type="checkbox"/>	<input type="checkbox"/>	AD_TLNMBR1	CHAR	30	0	First telephone no.: dialling code+number
PB_CHANGED_AT	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_CREATED_AT	DEC	15	0	Created at (Timestamp)
PB_CREATED_AT	<input type="checkbox"/>	<input type="checkbox"/>	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

Field	Key	Ini...	Data Element	Data Type	Length	Deci...	Short Description
MANDT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	MANDT	CLNT	3	0	Client
PB_CATEGORY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Z2812_PB_CATEG...	CHAR	2	0	Category of Contact Numbers
PB_SPRAS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	SPRAS	LANG	1	0	Language Key
PB_CATEGORY_DESC	<input type="checkbox"/>	<input type="checkbox"/>	Z2812_PB_CATEG...	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 Interface 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 Interface 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

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 00-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.

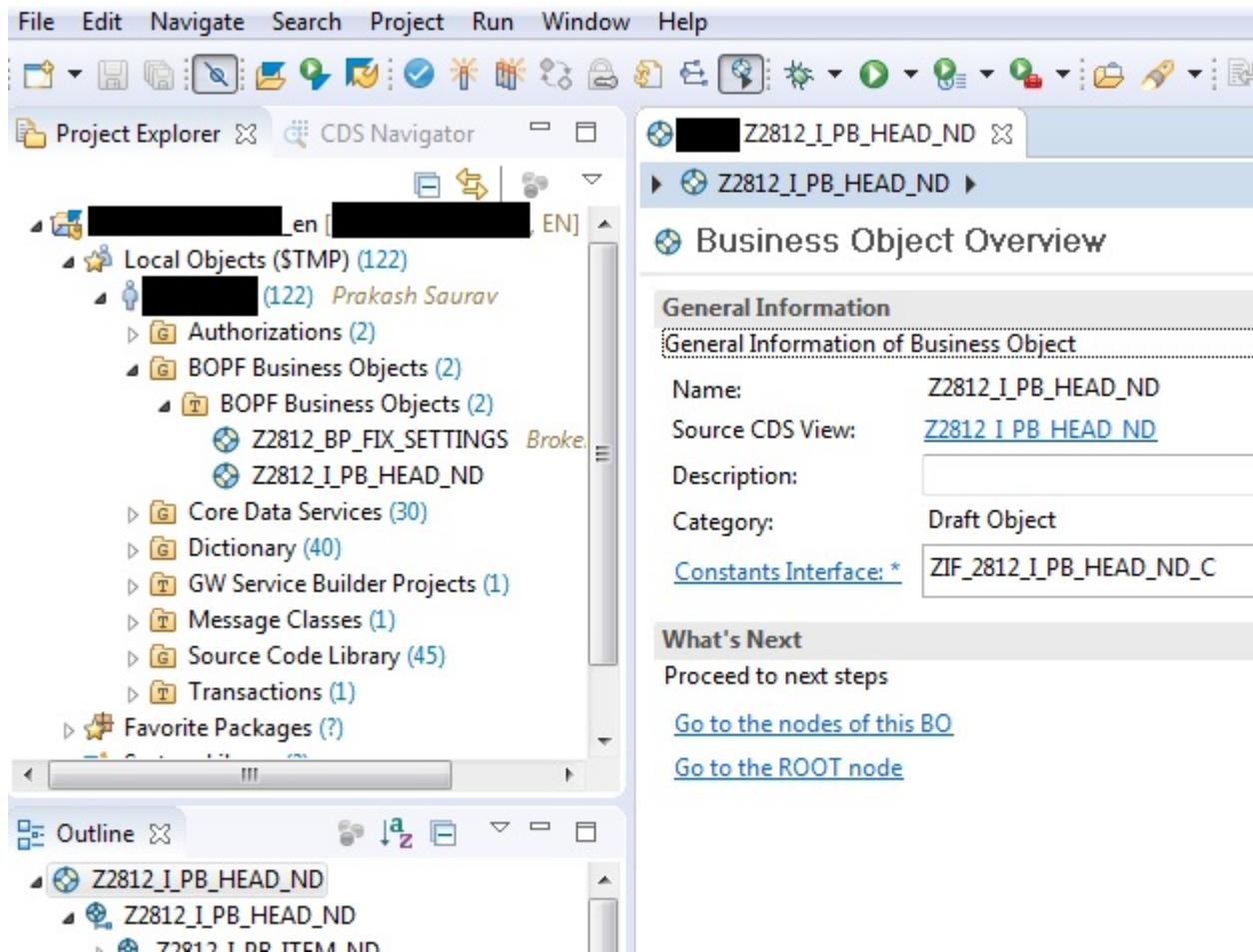
SOURCE: [Introduction to Business Object Processing Framework \(BOPF\)](#)

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 **B0B**, **B0BF**, **B0BX**. 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 B0 => Display Nodes in Hierarchy
- Go to the Root Node => Display Node and its Elements
 - [Overview](#)
 - [Alternative Keys](#)
 - [Properties](#)
 - [Associations](#)
 - [Actions](#)
 - [Determinations](#)
 - [Validations](#)
 - [Authorizations](#)
- [Display Generated Business Object using B0BF](#)



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.

Class Builder: Display Interface ZIF_2812_I_PB_HEAD_ND_C

Attribute	Level	R...	Typing	Associated Type	Description
SC_ACTION	Constant	<input type="checkbox"/>			Actions
SC_ACTION_ATTRIBUTE	Constant	<input type="checkbox"/>			Action Parameter Attributes
SC_ALTERNATIVE_KEY	Constant	<input type="checkbox"/>			Alternative Keys
SC_ASSOCIATION	Constant	<input type="checkbox"/>			Associations
SC_ASSOCIATION_ATTRIBUTE	Constant	<input type="checkbox"/>			Association Parameter Attributes
SC_BO_KEY	Constant	<input type="checkbox"/>			Bo Keys
SC_BO_NAME	Constant	<input type="checkbox"/>			Bo Names
SC_DETERMINATION	Constant	<input type="checkbox"/>			Determinations
SC_GROUP	Constant	<input type="checkbox"/>			Groups
SC_MODEL_VERSION	Constant	<input type="checkbox"/>			Model Versions
SC_NODE	Constant	<input type="checkbox"/>			Nodes

[<< 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.

Node Overview

General Information

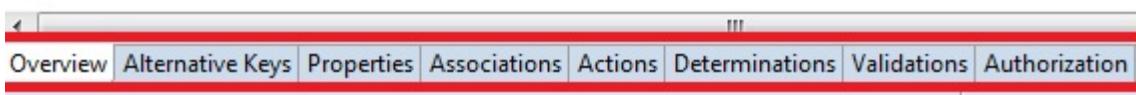
General Information of Node

Name: *	Z2812_I_PB_HEAD_ND		New...	Browse...
Source CDS View:	Z2812_I_PB_HEAD_ND		New...	Browse...
Description:				
Persistent Structure: *	ZS2812_I_PB_HEAD_ND_D		New...	Browse...
Transient Structure:			New...	Browse...
Combined Structure: *	ZS2812_I_PB_HEAD_ND			
Combined Table Type: *	ZT2812_I_PB_HEAD_ND			
Database Table: *	Z2812_PB_D_HDR_D			
<input checked="" type="checkbox"/> Create Enabled	<input checked="" type="checkbox"/> Update Enabled	<input checked="" type="checkbox"/> Delete Enabled		
<input type="checkbox"/> Text Node				

Implementation Details

Node Implementation Details of Draft Business Object

Draft Class:	/BOBF/CL_LIB_DR		New...	Browse...
--------------	-----------------	--	--------	-----------



[<< 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.

The screenshot shows a SAP interface for managing alternative keys. At the top, there are two blue navigation buttons labeled 'Z2812_I_PB_HEAD_ND'. Below them is a title 'Alternative Keys' with a key icon. Underneath is a toolbar with 'New...' and 'Delete' buttons. A search bar says 'type filter text'. A table lists one key: 'ACTIVE ENTITY KEY' with 'Uniqueness' set to 'Not unique'.

Name	Uniqueness
ACTIVE ENTITY KEY	Not unique

[<< 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.

Properties

type filter text

Element Name	Enabled	ReadOnly	Mandatory
KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PARENT_KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ROOT_KEY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
PB_ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
PB_OWNER	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
PB_FIRST_NAME	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PB_LAST_NAME	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PB_EMAIL_ID	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PB_CREATED_AT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
PB_CHANGED_AT	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ACTIVEUUID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
HASACTIVEENTITY	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
DRAFTENTITYCREATIONDATETIME	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DRAFTENTITYLASTCHANGEDATETIME	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DRAFTENTITYCONSISTENCYSTATUS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
ISACTIVEENTITY	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

[Overview](#) | [Alternative Keys](#) | [Properties](#) | [Associations](#) | [Actions](#) | [Determinations](#) | [Valid](#)

[<< Initial View](#)

Associations

Association to Item Data is generated as per CDS annotation.

Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND >

Associations

[New...](#) [Delete](#)

type filter text

Name	Multiplicity	Target Node	Implementation Class
_ITEMDATA	0..n	Z2812_I_PB_HEAD_ND~>Z2812_I_PB_ITEM_ND	

[<< 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.

Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND >

Actions

[New...](#) [Delete](#)

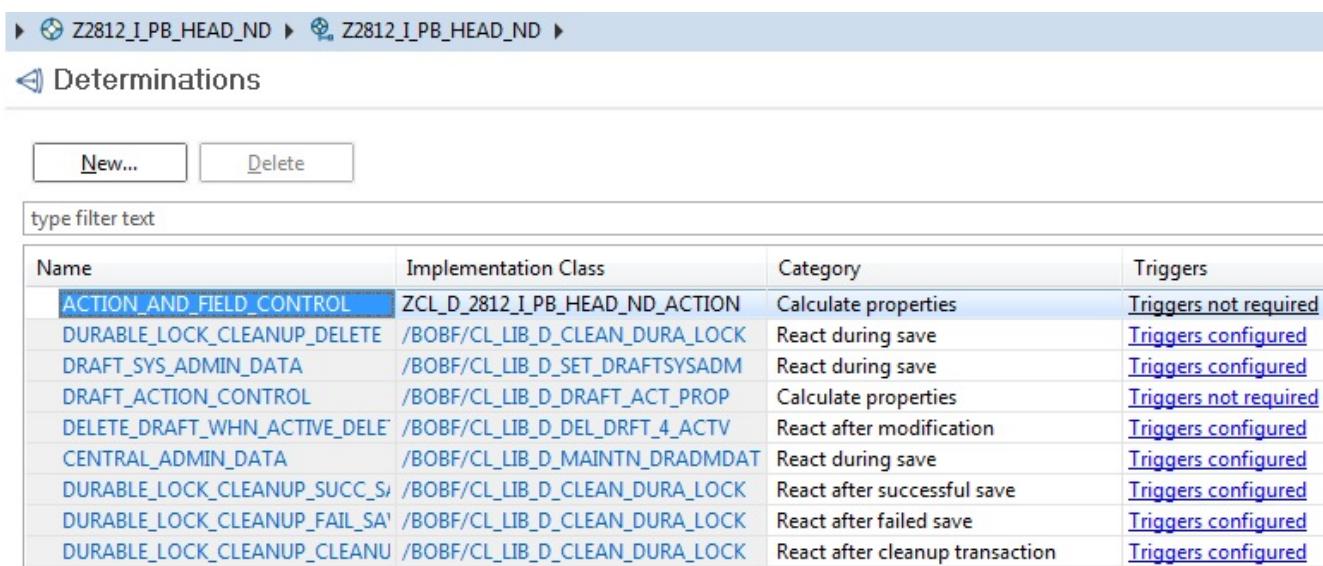
type filter text

Name	Implementation Class	Instance Multiplicity	Parameter Structure	Exporting Type
EDIT	/BOBF/CL_LIB_A_EDIT	Single Node Instance	/BOBF/S_LIB_A_DRAFT_EDIT_IMP	Node
ACTIVATION	/BOBF/CL_LIB_A_ACTIVATION	Single Node Instance		Node
VALIDATION	/BOBF/CL_LIB_A_VALIDATION	Single Node Instance	/BOBF/S_LIB_A_IN_DRAFT_VALIDATE	Type
PREPARATION	/BOBF/CL_LIB_A_PREPARATION	Single Node Instance	/BOBF/S_LIB_A_IN_DRAFT_PREPARE	Node

[<< Initial View](#)

Determinations

Determinations are tied to the event and are triggered implicitly. You can't 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.



The screenshot shows the SAP Fiori interface for managing Determinations. The top navigation bar has two entries: 'Z2812_I_PB_HEAD_ND' and 'Determinations'. Below the navigation is a toolbar with 'New...', 'Delete', and a search field labeled 'type filter text'. A table lists various determinations with columns: Name, Implementation Class, Category, and Triggers. The table rows are:

Name	Implementation Class	Category	Triggers
ACTION_AND_FIELD_CONTROL	ZCL_D_2812_I_PB_HEAD_ND_ACTION	Calculate properties	Triggers not required
DURABLE_LOCK_CLEANUP_DELETE	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React during save	Triggers configured
DRAFT_SYS_ADMIN_DATA	/BOBF/CL_LIB_D_SET_DRAFTSYSADM	React during save	Triggers configured
DRAFT_ACTION_CONTROL	/BOBF/CL_LIB_D_DRAFT_ACT_PROP	Calculate properties	Triggers not required
DELETE_DRAFT_WHN_ACTIVE_DELETE	/BOBF/CL_LIB_D_DEL_DRFT_4_ACTV	React after modification	Triggers configured
CENTRAL_ADMIN_DATA	/BOBF/CL_LIB_D_MAINTN_DRADMDAT	React during save	Triggers configured
DURABLE_LOCK_CLEANUP_SUCC_SAVE	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after successful save	Triggers configured
DURABLE_LOCK_CLEANUP_FAIL_SAVE	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after failed save	Triggers configured
DURABLE_LOCK_CLEANUP_CLEANUP	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after cleanup transaction	Triggers configured

[<< 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.

The screenshot shows a SAP Fiori application interface. The top navigation bar has two entries: 'Z2812_I_PB_HEAD_ND' and 'Validations'. Below the navigation is a toolbar with 'New...' and 'Delete' buttons, and a search input field labeled 'type filter text'. A table lists validation details:

Name	Implementation Class	Category	Triggers
DURABLE_LOCK_CREATE_FOR_NEW	/BOBF/CL_LIB_V_NEW_DURA_LOCK	Action Check	Triggers not configured

[<< Initial View](#)

Authorizations

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

The screenshot shows a SAP Fiori application interface. The top navigation bar has two entries: 'Z2812_I_PB_HEAD_ND' and 'Authorization'. Below the navigation is a section titled 'General Information' with a sub-section 'General Information of Authorization'. The configuration includes:

- Node Has Own Check
- Authorization Class**:
 Define Own Authorization Class
Authorization Class: ZCL_AU_2812_I_PB_HEAD_ND

Based on the requirement you can use method 'CHECK_STATIC_AUTHORITY' or 'CHECK_INSTANCE_AUTHORITY'

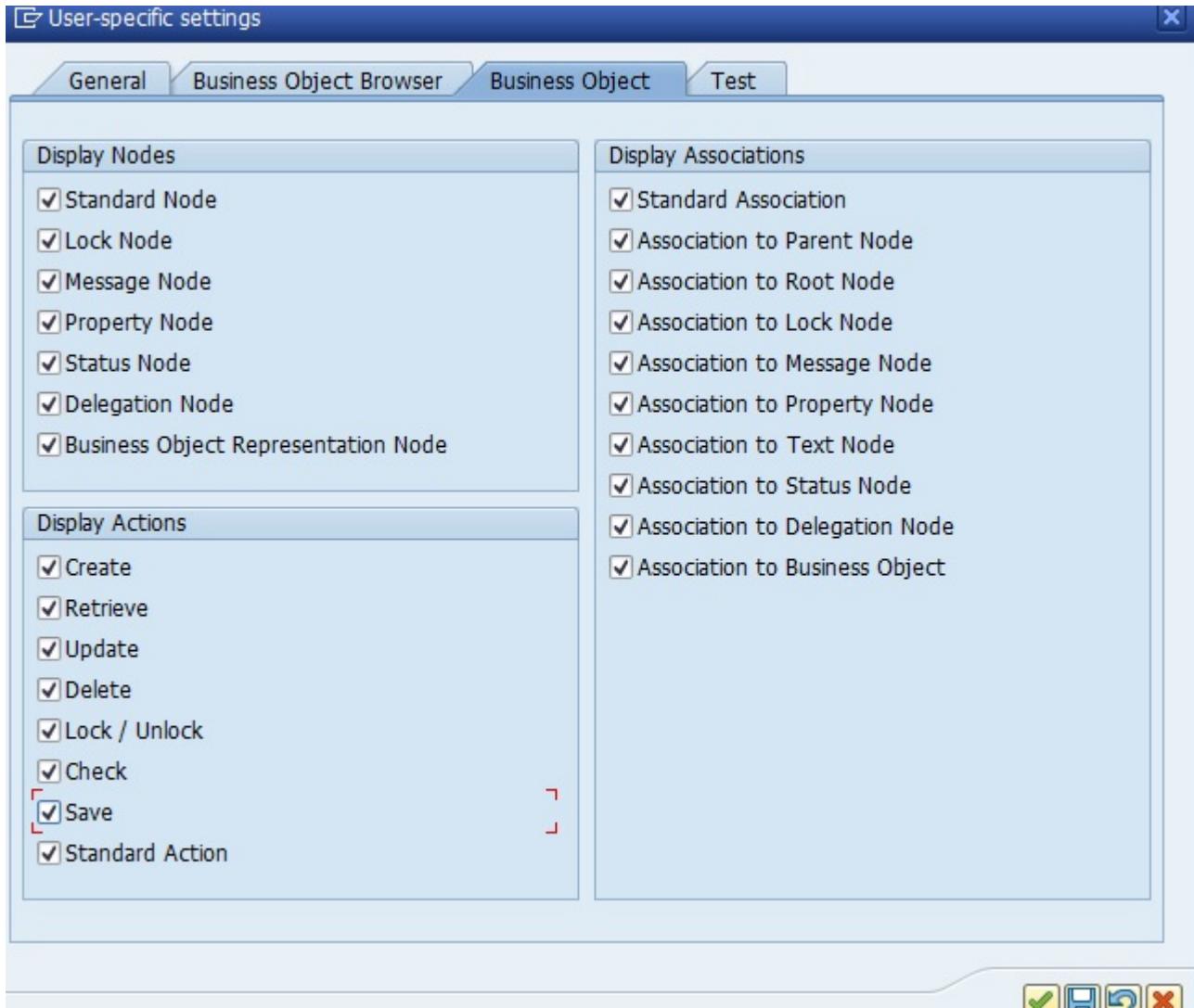
Class Builder: Display Class ZCL_AU_2812_I_PB_HEAD_ND

Method	Level	Visibility	M...	Description
/BOBF/IF_LIB_AUTH_DRAFT_ACTIVE~CHECK	Instance Method	Public		Executes the static authorization check
/BOBF/IF_LIB_AUTH_DRAFT_ACTIVE~CHE...	Instance Method	Public		Executes the instance-based authorization check
CHECK_AUTHORITY	Instance Method	Public		execute static and instance based authority check
CHECK_AUTHORITY_STATICALLY	Instance Method	Public		execute static authority check
GET_QUERY_CONDITION_PROVIDER	Instance Method	Public		Get auth. condition provider to execute query based on SADL
CONSTRUCTOR	Instance Method	Public		

[<< 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

The screenshot shows the 'Business Object Detail Browser' window. The tree view on the left shows the following structure:

- Z2812_I_PB_HEAD_ND
 - Node Structure
 - Z2812_I_PB_HEAD_ND
 - Z2812_I_PB_HEAD_ND_LOCK
 - Z2812_I_PB_HEAD_ND_MESSAGE
 - Z2812_I_PB_HEAD_ND_PROPERTY
 - Z2812_I_PB_ITEM_ND

Node Elements

Business Object Detail Browser	Description
Node Elements	
○ Z2812_I_PB_HEAD_ND	
▼ Node Categories	
• ROOT	
▼ Associations	
• LOCK	
• MESSAGE	
• PROPERTY	
▼ _ITEMDATA	Composition to Z2812_I_PB_ITEM_ND
• Association Parameter Value	
▼ Determinations	
• ACTION_AND_FIELD_CONTROL	Generated
• CENTRAL_ADMIN_DATA	Generated for Draft Object
• DELETE_DRAFT_WHN_ACTIVE	Generated for Draft Object
• DRAFT_ACTION_CONTROL	Generated for Draft Object
• DRAFT_SYS_ADMIN_DATA	Generated for Draft Object
• DURABLE_LOCK_CLEANUP_CLEA	Generated for Draft Object
• DURABLE_LOCK_CLEANUP_DELETE	Generated for Draft Object
• DURABLE_LOCK_CLEANUP_FAIL	Generated for Draft Object
• DURABLE_LOCK_CLEANUP_SUCCESS	Generated for Draft Object
▼ Validations	
• DURABLE_LOCK_CREATE_FOR_I	Generated for Draft Object

Node Elements Image 1

Node Elements	
○ Z2812_I_PB_HEAD_ND	
► Node Categories	
► Associations	
► Determinations	
► Validations	
▼ Actions	
► ACTIVATION	Generated for Draft Object
► CREATE_Z2812_I_PB_HEAD_ND	
► DELETE_Z2812_I_PB_HEAD_ND	
► EDIT	Generated for Draft Object
► LOCK_Z2812_I_PB_HEAD_ND	
► PREPARATION	Generated for Draft Object
► SAVE_Z2812_I_PB_HEAD_ND	
► UNLOCK_Z2812_I_PB_HEAD_ND	
► UPDATE_Z2812_I_PB_HEAD_ND	
► VALIDATE_Z2812_I_PB_HEAD_I	
► VALIDATION	Generated for Draft Object
• Queries	
▼ Alternative Keys	
• ACTIVE_ENTITY_KEY	Generated for Draft Object
▼ Status Variables	
• DRAFT_CONSISTENCY_STATUS	
• Attribute Value Sets	

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
-

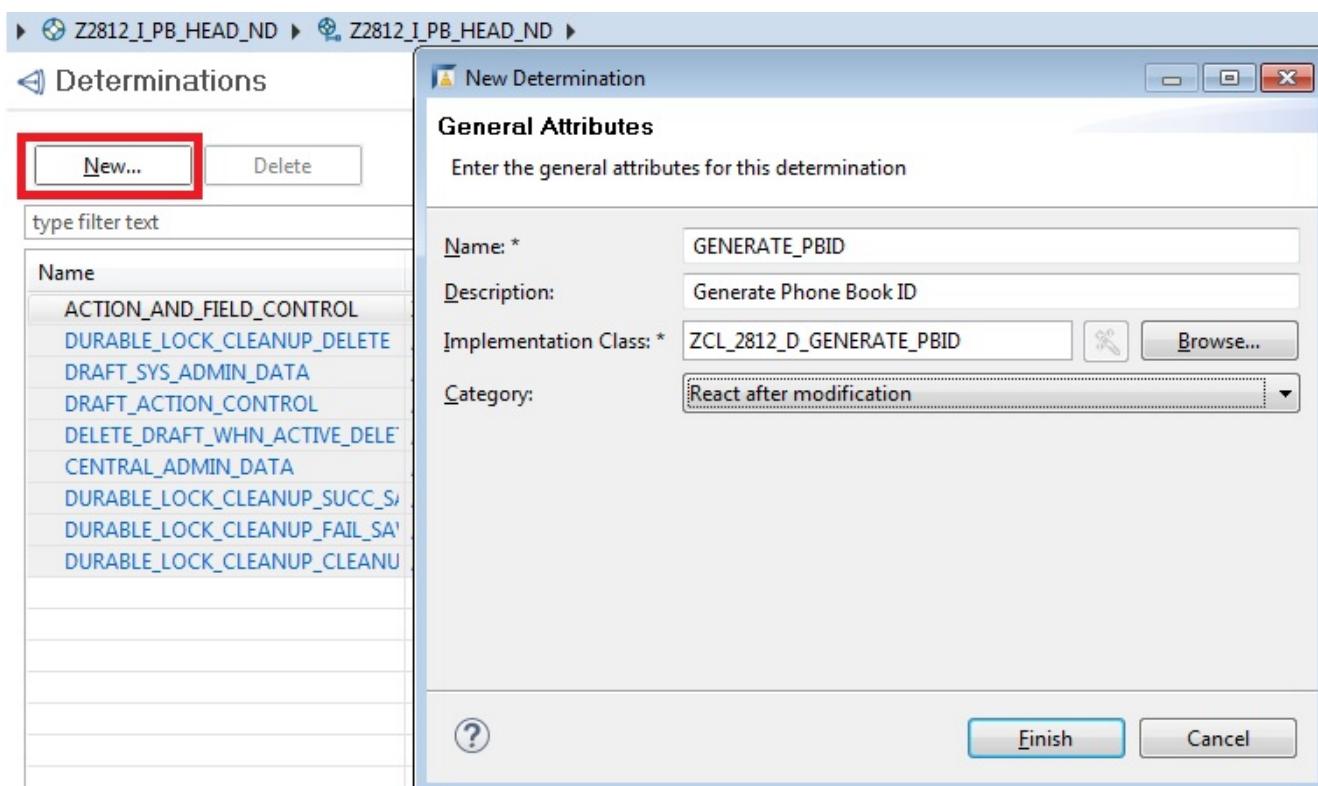
Determinations

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.

Z2812_I_PB_HEAD_ND ▶ Z2812_I_PB_HEAD_ND ▶

Determinations

New... Delete

type filter text

Name	Implementation Class	Category	Triggers
ACTION_AND_FIELD_CONTROL	ZCL_D_2812_I_PB_HEAD_ND_ACTION	Calculate properties	Triggers not required
DURABLE_LOCK_CLEANUP_DELETE	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React during save	Triggers configured
DRAFT_SYS_ADMIN_DATA	/BOBF/CL_LIB_D_SET_DRAFTSYSADM	React during save	Triggers configured
DRAFT_ACTION_CONTROL	/BOBF/CL_LIB_D_DRAFT_ACT_PROP	Calculate properties	Triggers not required
DELETE_DRAFT_WHN_ACTIVE_DELETE	/BOBF/CL_LIB_D_DEL_DRFT_4_ACTV	React after modification	Triggers configured
CENTRAL_ADMIN_DATA	/BOBF/CL_LIB_D_MAINTN_DRADMDAT	React during save	Triggers configured
DURABLE_LOCK_CLEANUP_SUCC_S	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after successful save	Triggers configured
DURABLE_LOCK_CLEANUP_FAIL_SA	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after failed save	Triggers configured
DURABLE_LOCK_CLEANUP_CLEANU	/BOBF/CL_LIB_D_CLEAN_DURA_LOCK	React after cleanup transaction	Triggers configured
GENERATE_PBID	ZCL_2812_D_GENERATE_PBID	React after modification	Triggers configured

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.

Z2812_I_PB_HEAD_ND ▶ Z2812_I_PB_HEAD_ND ▶ GENERATE_PBID

Determination Overview

General Information

General Information of Determination

Name:*	GENERATE_PBID
Description:	Generate PhoneBook ID & Update AdminData
Implementation Class: *	ZCL_2812_D_GENERATE_PBID
Category:	React after modification

Triggers

Information on Triggers

New Delete

type filter text

Node	Association	Create	Update
Z2812_I_PB_HEAD_ND		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

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.*

Node	Determination	Dependency

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

Method	Level	Visibility	M...	Description
/BOBF/IF_FRW_DETERMINATION~CHECK_DEI	Instance	Method	Public	Check for Relevant Data Changes
/BOBF/IF_FRW_DETERMINATION~CHECK	Instance	Method	Public	Check Values of Relevant Fields
/BOBF/IF_FRW_DETERMINATION~EXECUTE	Instance	Method	Public	Perform Determination

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

```

*-----*
* Method Name : /B0BF/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 : 
/B0BF/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*.

The screenshot shows the SAP Business Object interface with the title bar [ER9] Z2812_I_PB_HEAD_ND. Below the title bar, there is a breadcrumb navigation path: Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND. A red box highlights the 'Determinations' link in the breadcrumb. On the left, there is a sidebar with tabs: New..., Delete, type filter text, Name, ACTION_AND_FIELD_C, DURABLE_LOCK_CLEANUP, and DRAFT SYS ADMIN D. The main content area is titled 'Determinations' and shows a list of determinations. One determination, 'Z2812_I_PB_ITEM_ND', is highlighted with a red box. To its right is a list of actions: ACTIVE_ENTITY_KEY : Not unique, _ITEMDATA -> Z2812_I_PB_HEAD_ND~>Z2812_I_PB_ITEM_ND, ACTIVATION : Single Node Instance, COPY : Multiple Node Instances, EDIT : Single Node Instance, EMAIL : Multiple Node Instances, PREPARATION : Single Node Instance, and VALIDATION : Single Node Instance. To the right of the list, there are four columns: Triggers (empty), Triggers not (empty), Triggers cor (empty), and Triggers cor (empty).

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**.

The screenshot shows the SAP Business Object interface with the title bar [ER9] Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND > Z2812_I_PB_ITEM_ND > UPDATE_I_ADMIN. A red box highlights the breadcrumb path. Below the title bar, there is a 'Determination Overview' header with a 'Go to the determination' link. The page is divided into sections: 'General Information' and 'Triggers'. In the 'General Information' section, there is a table with the following fields: Name: * UPDATE_I_ADMIN, Description: Update Item Admin Data, Implementation Class: * ZCL_2812_D_UPDATE_I_ADMIN, and Category: React after modification. In the 'Triggers' section, there is a table with the following fields: Node: Z2812_I_PB_ITEM_ND, Association: (empty), Create: checked, Update: checked, Delete: unchecked, Load: unchecked, and Determine: unchecked.

Business Logic

```
*-----*  
* Method Name :  
/B0BF/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](#)

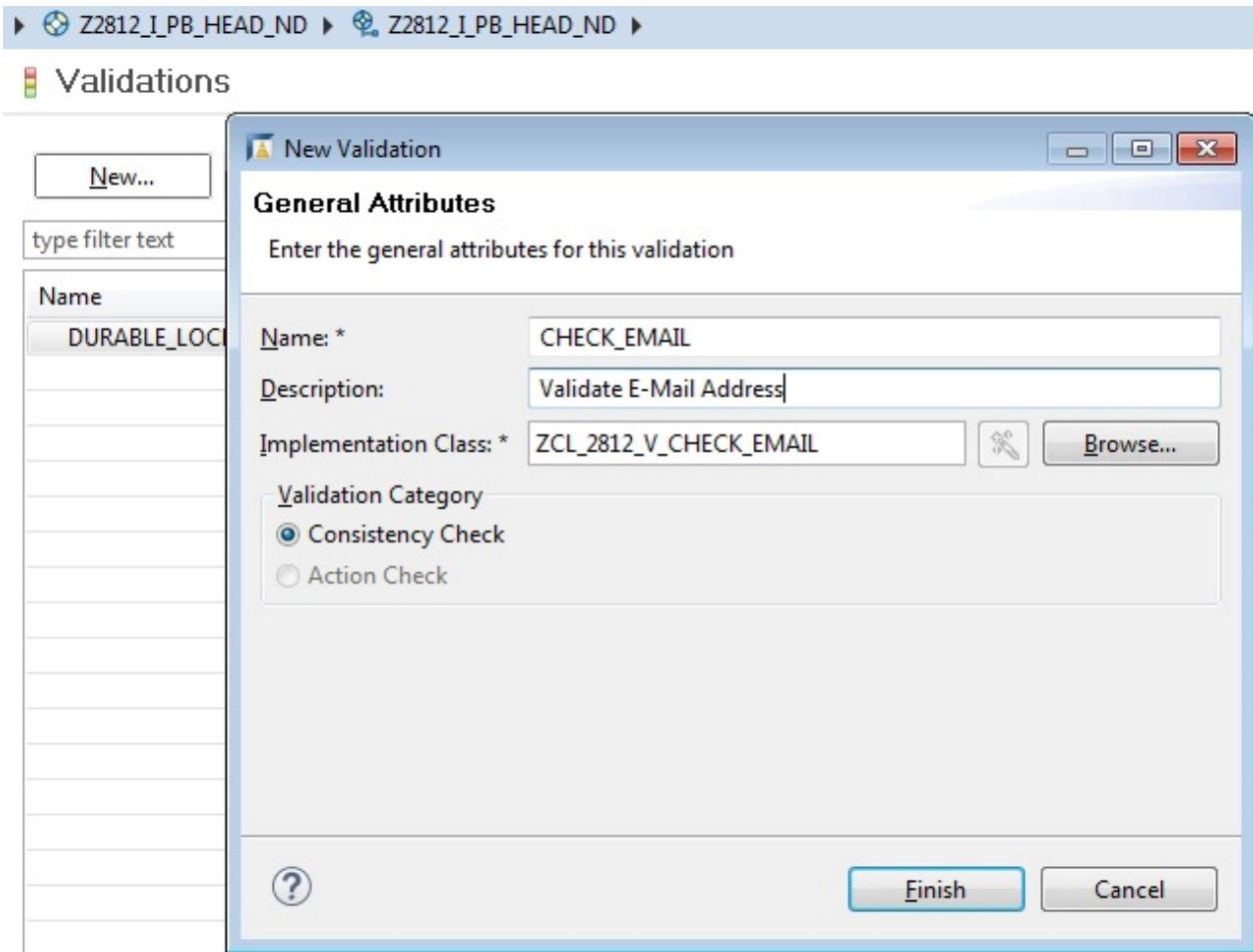
Validations

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.

Validation Overview

General Information

General Information of Validation

Name:*	CHECK_EMAIL
Description:	Validate E-Mail Address
Implementation Class: *	ZCL_2812_V_CHECK_EMAIL
Category:	Consistency Check

Triggers

Information on Triggers

[New](#)

[Delete](#)

type filter text

Node	Association	Create	Update	Delete
Z2812_I_PB_HEAD_ND		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

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

Class Builder: Display Class ZCL_2812_V_CHECK_EMAIL

Method	Level	Visibility	M...	Description
/BOBF/IF_FRW_VALIDATION-CHECK_DELTA	Instance Method	Public		Check for Relevant Data Changes
/BOBF/IF_FRW_VALIDATION-CHECK	Instance Method	Public		Check Values of Relevant Fields
/BOBF/IF_FRW_VALIDATION-EXECUTE	Instance Method	Public		Execute a Validation

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

*-----

```

-----*
* Method Name          : /B0BF/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 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**.

Validation Overview

General Information

General Information of Validation

Name:*	<input type="text" value="CHECK_TELENO"/>
Description:	<input type="text" value="Validate Telephone Number"/>
Implementation Class: *	<input type="text" value="ZCL_2812_V_CHECK_TELENO"/>
Category:	<input type="text" value="Consistency Check"/>

Triggers

Information on Triggers

[New](#) [Delete](#)

type filter text

Node	Association	Create	Update	Delete
Z2812_I_PB_ITEM_ND		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Validations

[New...](#) [Delete](#)

type filter text

Name	Implementation Class	Category	Triggers
CHECK_CATG	ZCL_2812_V_CHECK_CATG	Consistency Check	Triggers configured

Business Logic

```

*-----
*-----*
* Method Name          : /B0BF/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 Action 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.

Display Business Object Z2812_I_PB_HEAD_ND, Active Version

Group	Z2812_I_PB_HEAD_ND
Description	SAVE_PREVENTION
Group Settings	
Group Category	Consistency Group
Node	Nothing chosen
Action	Nothing chosen
Status Variable	Nothing chosen

Validations from both nodes are assigned to same consistency group.

Display Business Object Z2812_I_PB_HEAD_ND, Active Version

Group Assignment	Description
Z2812_I_PB_HEAD_ND	SAVE_PREVENTION
Validations	
• <input checked="" type="checkbox"/> CHECK_EMAIL	Validate E-Mail Address
• <input checked="" type="checkbox"/> CHECK_CATG	Validate Category
• <input checked="" type="checkbox"/> CHECK_TELENO	Validate Telephone Number
Delegated Nodes	

[<< Development List](#)

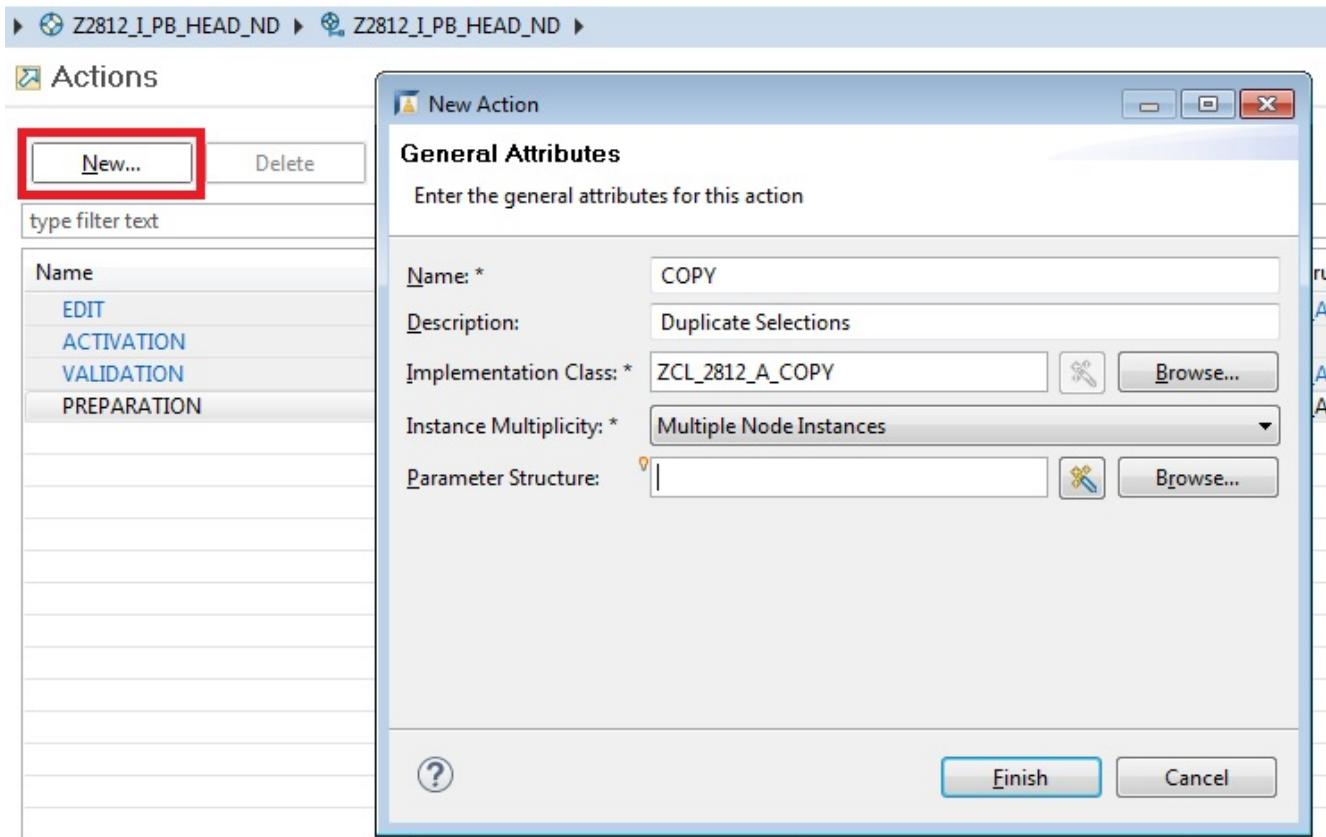
Actions

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



Actions			
Name	Implementation Class	Instance Multiplicity	Parameter Structure
EDIT	/BOBF/CL_LIB_A_EDIT	Single Node Instance	/BOBF/S_LIB_A_DRAFT_EDIT_IMP
ACTIVATION	/BOBF/CL_LIB_A_ACTIVATION	Single Node Instance	
VALIDATION	/BOBF/CL_LIB_A_VALIDATION	Single Node Instance	/BOBF/S_LIB_A_IN_DRAFT_VALIDAT
PREPARATION	/BOBF/CL_LIB_A_PREPARATION	Single Node Instance	/BOBF/S_LIB_A_IN_DRAFT_PREPARE
COPY	ZCL_2812_A_COPY	Multiple Node Instances	
EMAIL	ZCL_2812_A_EMAIL	Multiple Node Instances	Z2812_S_PB_EMAIL

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.

Class Builder: Display Class ZCL_2812_A_COPY

Method	Level	Visibility	Description
/BOBF/IF_FRW_ACTION~EXECUTE	Instance Method	Public	Carries Out an Action
/BOBF/IF_FRW_ACTION~PREPARE	Instance Method	Public	Manipulate Object List Before Validation
/BOBF/IF_FRW_ACTION~RETRIEVE_DEFAULT	Instance Method	Public	Gives the default parameters of an action

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

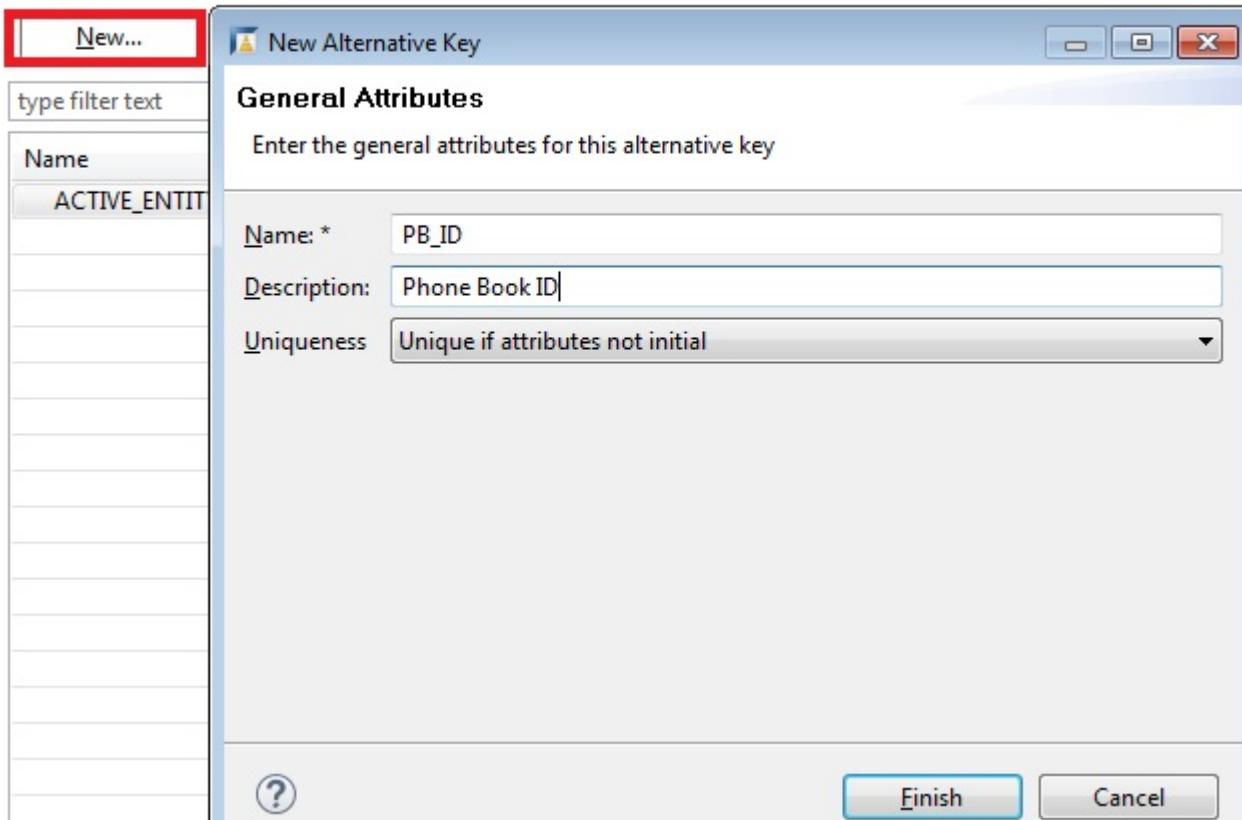
This key will allow us to find node instance using Phone Book ID.

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

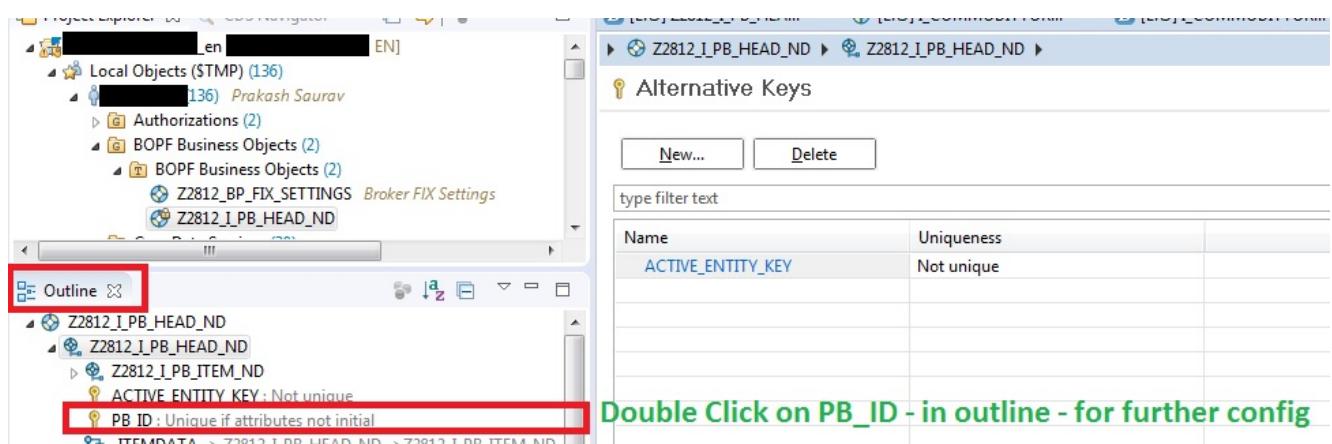
Alternative Keys Tab at the bottom or Click on Alternative Keys Link on Side.

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

🔑 Alternative Keys



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

Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND > PB_ID

Alternative Key Overview

[Go to the alternative keys](#)

General Information

General Information of Alternative Key

Name: *	PB_ID
Description:	Phone Book ID
Uniqueness:	Unique if attributes not initial Uniqueness Check <input checked="" type="radio"/> Check After Modify <input type="radio"/> No Check

Key Fields

List of Key Fields

Field Name	Add...
PB_ID	Remove

Data Type Details

Data Type details of selected Key Field

Data Type: *	Z2812_PHONEBOOK_ID	
Data Table Type: *	Z2812_T_K_PBID	

[<< Development List](#)

[<< Top](#)

Test BOPF object using BOBT

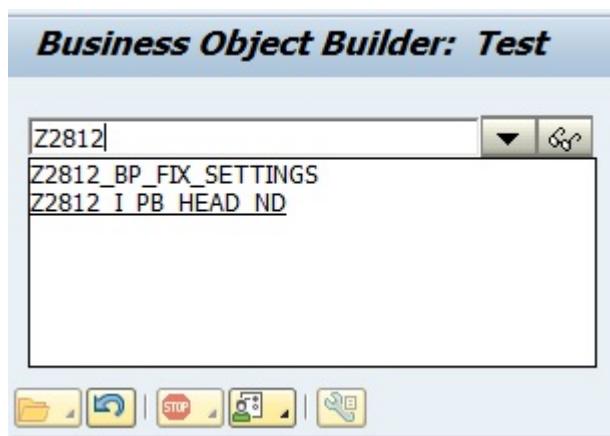
1. [Design / Prototype](#)
2. [Create Dictionary Objects](#)
3. [Create Interface View \(CDS\)](#)
4. [Business Object Generation \(CDS – Object Model Annotations \)](#)

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.

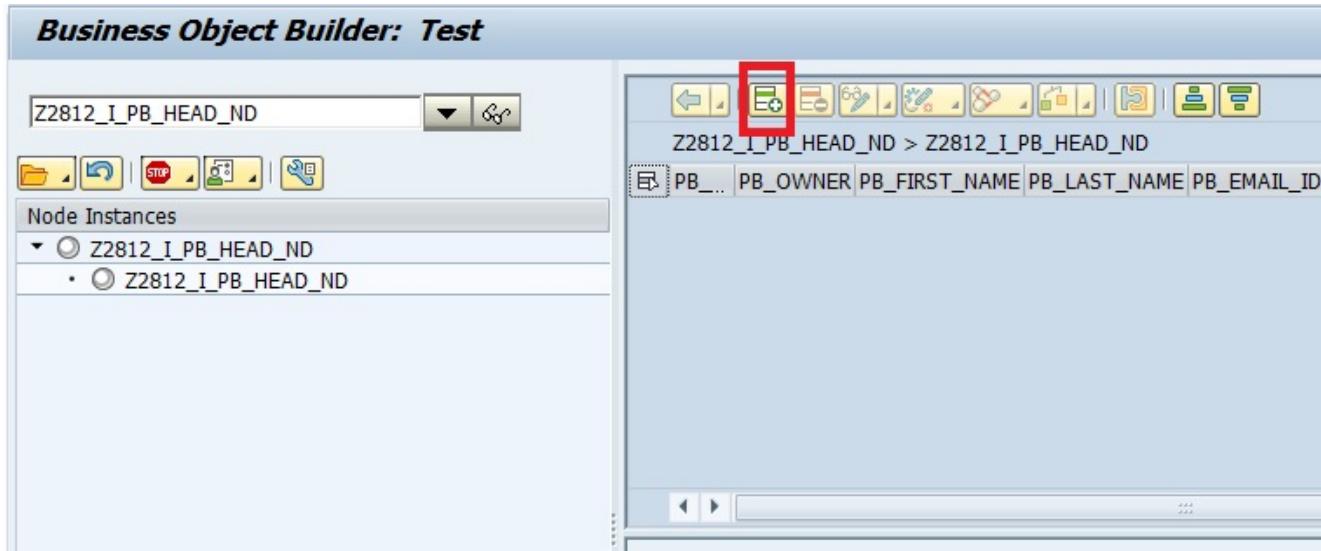
Step 1: Go to Transaction ‘**BOBT**’

Step 2: Enter the Business Object name as shown below

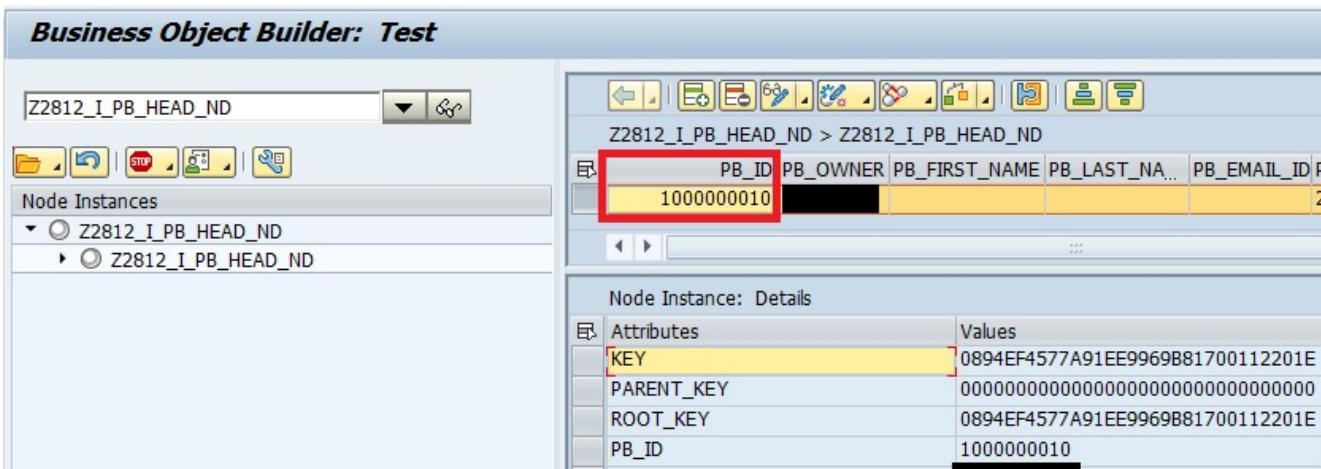


Step 3: Click on ‘New Entry’ to create new instance of the

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.

Business Object Builder: Test

The screenshot shows the Business Object Builder interface with the title "Business Object Builder: Test". In the left pane, there is a tree view under "Node Instances" with two items: "Z2812_I_PB_HEAD_ND" and "Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND". The right pane displays a table with columns: PB_ID, PB_OWNER, PB_FIRST_NAME, PB_LAST_NAME, PB_EMAIL_ID, PB_CREATED_AT, and PB_CHANGED_AT. A row is selected with PB_ID = 1000000010, PB_OWNER = [REDACTED], PB_FIRST_NAME = John, PB_LAST_NAME = Smith, PB_EMAIL_ID = john.smith, PB_CREATED_AT = 20190407022252, and PB_CHANGED_AT = 20190407022252. The "PB_EMAIL_ID" cell is highlighted with a red border. Below the table, a "Node Instance: Details" panel shows attribute values for the selected row. At the bottom, a "Messages" panel contains a single message: "Type Text: Invalid E-Mail, correct format is 'local-part@domain' e.g. 'abc@xyz.com' Node Name: Z2812_I_PB_HEAD_ND Z2812_I_PB_HEAD_ND".

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

The screenshot shows the Business Object Builder interface with the title "Business Object Builder: Test". In the left pane, the tree view shows "Z2812_I_PB_HEAD_ND" and "Z2812_I_PB_HEAD_ND > Z2812_I_PB_HEAD_ND > _ITEMDATA". The right pane displays a table with columns: PB_ID, PB_OWNER, PB_FIRST_NAME, PB_LAST_NAME, PB_EMAIL_ID, PB_CREATED_AT, and PB_CHANGED_AT. A row is selected with PB_ID = 1000000010, PB_OWNER = [REDACTED], PB_FIRST_NAME = John, PB_LAST_NAME = Smith, PB_EMAIL_ID = john.smith@example.com, PB_CREATED_AT = 20190407.02..., and PB_CHANGED_AT = 20190407.023... . The "PB_EMAIL_ID" cell is highlighted with a red border. Below the table, a "Node Instance: Details" panel shows attribute values for the selected row. At the bottom, a "Messages" panel contains a single message: "Type Text: Invalid phone number, correct format is '(XXX) XXX-XXXX' e.g. '(555) 555-1234' Node Name: Z2812_I_PB_HEAD_ND Z2812_I_PB_HEAD_ND > _ITEMDATA".

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.

Business Object Builder: Test

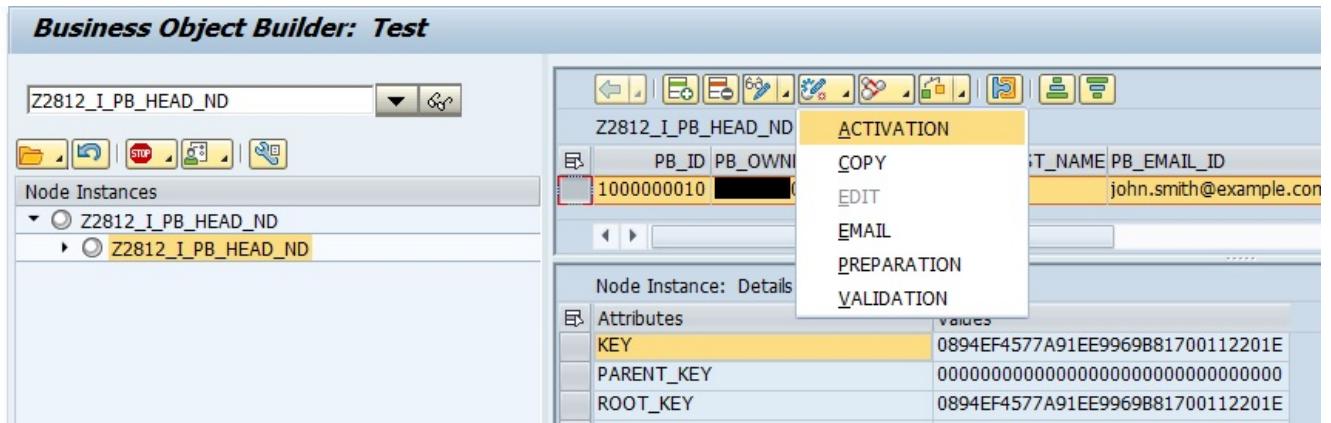
The screenshot shows the Business Object Builder interface with the title "Business Object Builder: Test". In the top right, there is a table with columns: PB_CATEGORY, PB_CATEGORY_DESC, PB_TELEPHONE, PB_CREATED_AT, PB_CHANGED_AT, ACTIVEUUID, and HASACTIVEENTITYD. A row is selected with values: HO, 91, and various timestamps. Below the table is a "Node Instance: Details" section with a table of attributes and their values. At the bottom, a "Messages" section contains an error message: "Type Text: Invalid Telephone#, correct format is '+Code-Number' e.g. '+91-9087654321' Node Name: Z2812_I_PB_HEAD_ND Z2812_I_PB_ITEM_ND".

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

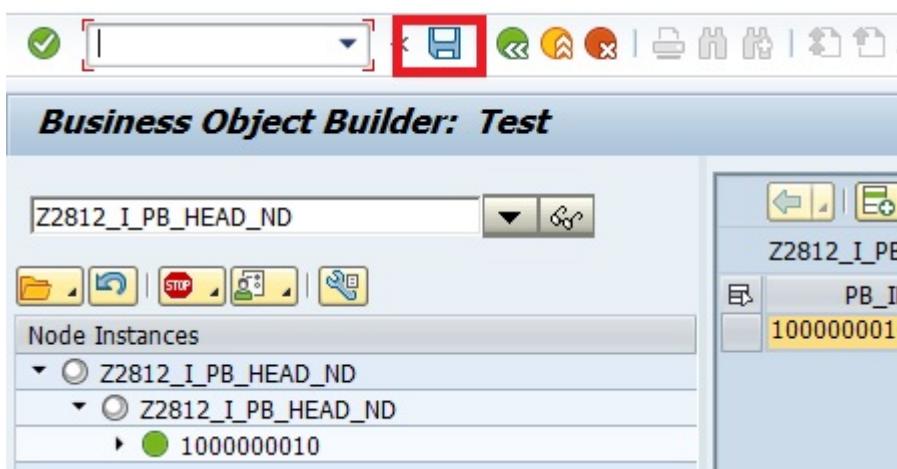
Business Object Builder: Test

The screenshot shows the Business Object Builder interface with the title "Business Object Builder: Test". The left pane shows a node tree with nodes: Z2812_I_PB_HEAD_ND, Z2812_I_PB_HEAD_ND, 1000000010, and _ITEMDATA. The node "1000000010" is highlighted with a red box. The right pane shows a table with columns: PB_CATEGORY, PB_CATEGORY_DESC, PB_TELEPHONE, and PB_CREATED_AT. A row is selected with values: HO, +91-9807654321, and 20.190.407.023. The table header "Z2812_I_PB_HEAD_ND : Z2812_I_PB_HEAD" is also highlighted with a red box.

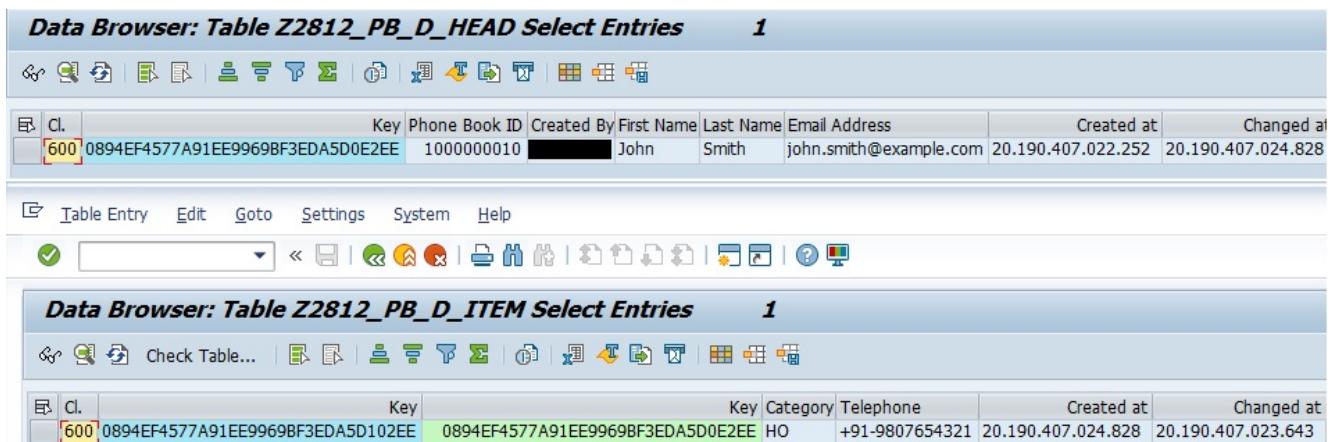
Step 8: Select the node and Activate it. It will prepare the node for persistence.



Step 9: Save the transaction to persist the entry.

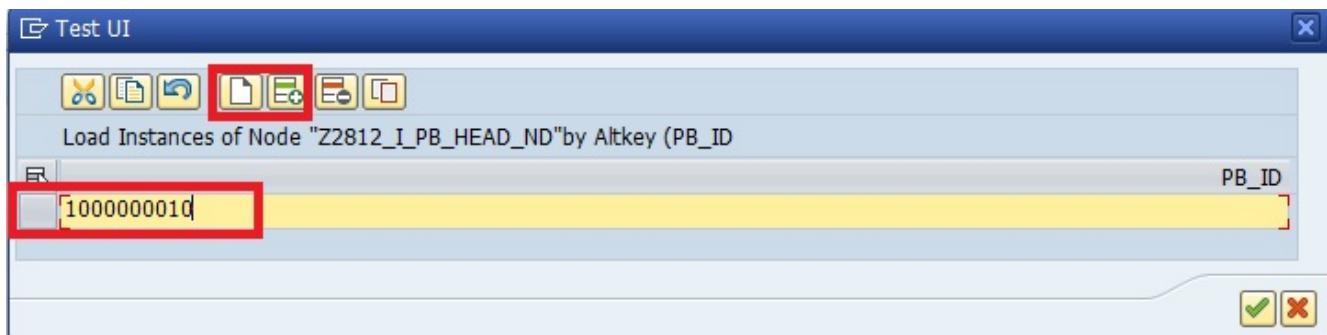
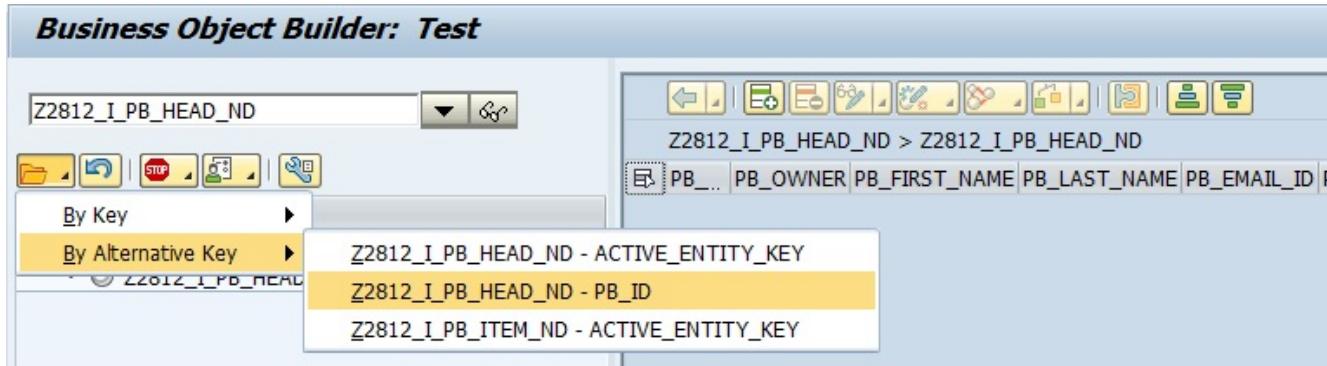


Step 10: Verify the entries persisted in database.

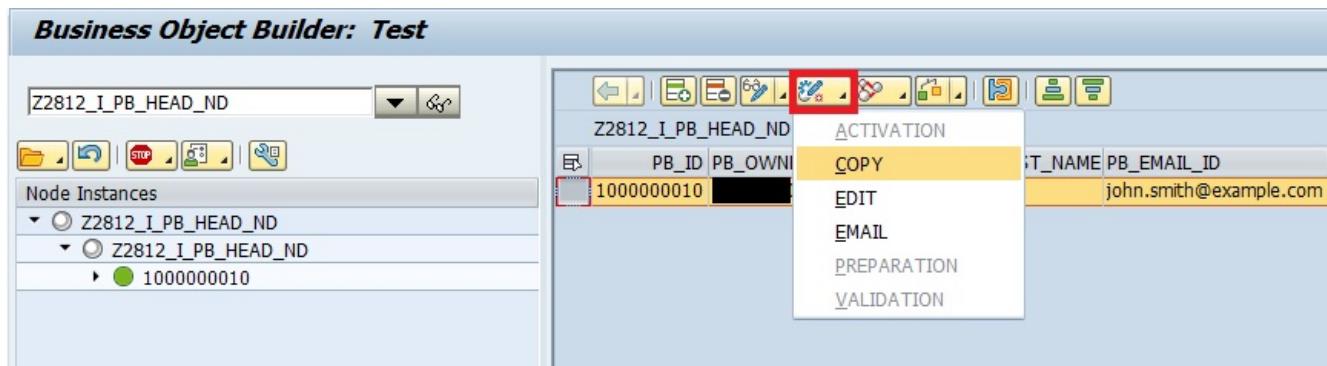


Step 11: Go to BOBT and open the Business Object, then use the

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	
	10000000011	C5275410	Thor	Lord of Thunder	lord.thunder@asgard.universe	20.190.407.030.210	20.190.407.030.440	0

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)

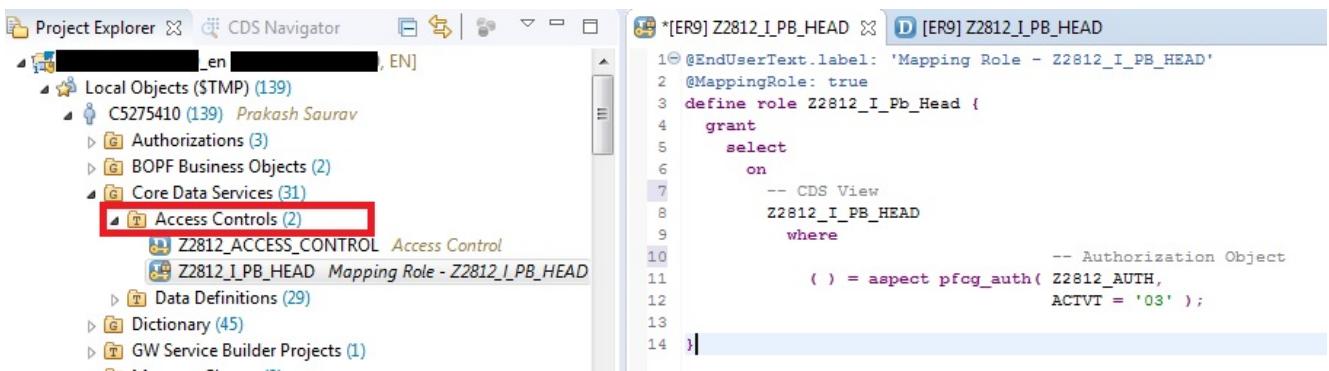
1. [Design / Prototype](#)
2. [Create Dictionary Objects](#)
3. [Create Interface View \(CDS\)](#)
4. [Business Object Generation \(CDS – Object Model Annotations \)](#)
5. [Generated Business Object](#)
6. [BOPF – Development](#)
 - [Determination](#)
 - [Validation](#)
 - [Action](#)
 - [Alternative 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](#)

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).



```

1@EndUserText.label: 'Mapping Role - Z2812_I_PB_HEAD'
2@MappingRole: true
3define role Z2812_I_Pb_Head {
4    grant
5        select
6        on
7            -- CDS View
8            Z2812_I_PB_HEAD
9            where
10           -- Authorization Object
11           ( ) = aspect pfccg_auth( Z2812_AUTH,
12                                     ACTVT = '03' );
13
14

```

```

@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](#)

Contact : Initial Screen

Selection Criteria

Phone Book ID:

000000000000



Initial Screen

Contact : Initial Screen

Selection Criteria

Phone Book ID:

100



10000000011

10000000010

Search Results

Phone Boo...	First Name	Last Name
10000000000	Friend Name1	Friend Name2
10000000010	John	Smith
10000000011	Thor	Lord of Thunder
10000000012	Ram	Naam
10000000013	oData	Create

[More Values...](#)

Dynamic Search & History Enabled

The screenshot shows a SAP Fiori application titled "Contact : Overview". It has two main sections: "Header" and "Item".

Header:

Full Name:	Thor Lord of Thunder
Email Address:	lord.thunder@asgard.universe
Created at:	07.04.2019 03:02:10
Changed at:	07.04.2019 03:04:40

Item:

Category	Category Description	Telephone
HO	Home	+91-9807654321

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)

The screenshot shows the initial list screen of the application. At the top, there is a SAP logo and a title "Manage Contacts". Below the title, there is a search bar with a placeholder "Search" and a dropdown menu set to "All". To the right of the search bar are buttons for "Adapt Filters (1)" and "Go". Underneath the search area, there is a table header with columns: "Phone Book ID", "Full Name", "Email Address", "Created at", and "Changed at". A message "To start, set the relevant filters." is displayed below the table header.

Initial List Screen (Click on Go to Load Data)

The screenshot shows a list report for contacts. The table has columns: "Phone Book ID", "Full Name", "Email Address", "Created at", and "Changed at". There are seven entries in the table. The first entry is "1000000013" with "oData Create" as the full name and "email.created@odata.request" as the email address. The fifth entry is "1000000010" with "John Smith" as the full name and "john.smith@example.com" as the email address. The sixth entry is "1000000007" with "Draft" as the full name and "1000000007" as the Phone Book ID. The last entry is "1000000012" with "Ram Naam" as the full name and "ram@naam.com" as the email address. Each row has a "Delete" button, a plus sign for creating, and a gear icon for settings.

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

The screenshot shows the detail view for contact "1000000018". The contact information is as follows:

General Information	Personal Data	Admin Data
Phone Book ID: 1000000018	First Name: EPRS	Created at: Apr 24, 2019, 2:47:06 PM
	Last Name: BLOG	Changed at: Apr 24, 2019, 2:47:06 PM
	Email Address: admin@erps.onlin	

Below this, there is a section for "Phone/Cell Numbers" with a table:

Category	Telephone	Changed at	Created at
○		Apr 24, 2019, 2:48:21 PM	Apr 24, 2019, 2:48:21 PM

New Phone Book ID is assigned and data can be filled

Select: Category

Hide Advanced Search Go

Category: Category

Category Desc.: Category Desc.

Items

	C	▲		Category Desc.
				HO Home
				MO Mobile
				OF Office
				WO Work

Value Help Supplied for Category

1000000018

General Information Phone/Cell Numbers

General Information

Phone Book ID:
1000000018

Phone/Cell Numbers

Category

MO Category 923

Messages

! Invalid Telephone#,correct format is '+Code-Number'
e.g. '+91-9087654321'

Admin Data

Created at:
Apr 24, 2019, 2:47:06 PM

Changed at:
Apr 24, 2019, 2:48:21 PM

Close Changed at
Apr 24, 2019, 2:48:21 PM

Validations are Working

1000000018

General Information		Phone/Cell Numbers									
General Information Phone Book ID: 1000000018		Personal Data First Name: EPRS Last Name: BLOG Email Address: admin@erps.onlin									
		Admin Data Created at: Apr 24, 2019, 2:47:06 PM Changed at: Apr 24, 2019, 2:48:21 PM									
Phone/Cell Numbers <table border="1"> <thead> <tr> <th>Category</th> <th>Telephone</th> <th>Changed at</th> <th>Created at</th> </tr> </thead> <tbody> <tr> <td>MO</td> <td>+91-9345678123</td> <td>Apr 24, 2019, 2:48:21 PM</td> <td>Apr 24, 2019, 2:48:21 PM ></td> </tr> </tbody> </table>				Category	Telephone	Changed at	Created at	MO	+91-9345678123	Apr 24, 2019, 2:48:21 PM	Apr 24, 2019, 2:48:21 PM >
Category	Telephone	Changed at	Created at								
MO	+91-9345678123	Apr 24, 2019, 2:48:21 PM	Apr 24, 2019, 2:48:21 PM >								
Delete + Edit											
Draft saved Save Cancel											

Save

Manage Contacts					
Standard					
Editing Status:					
Search					
Phone Book ID	Full Name	Email Address	Created at	Changed at	
1000000018	EPRS BLOG	admin@erps.onlin	Apr 24, 2019, 2:47:06 PM	Apr 24, 2019, 2:59:37 PM >	
1000000013	oData Create	email.created@odata.request	Apr 15, 2019, 4:38:41 PM	Apr 15, 2019, 4:38:41 PM >	

New Contact is Saved

1000000018 EPRS BLOG

General Information		Phone/Cell Numbers									
General Information Phone Book ID: 1000000018		Personal Data First Name: EPRS Last Name: BLOG Email Address: admin@erps.onlin									
		Admin Data Created at: Apr 24, 2019, 2:47:06 PM Changed at: Apr 24, 2019, 2:59:37 PM									
Phone/Cell Numbers <table border="1"> <thead> <tr> <th>Category</th> <th>Telephone</th> <th>Changed at</th> <th>Created at</th> </tr> </thead> <tbody> <tr> <td>Mobile (MO)</td> <td>+91-9345678123</td> <td>Apr 24, 2019, 2:48:21 PM</td> <td>Apr 24, 2019, 2:59:37 PM ></td> </tr> </tbody> </table>				Category	Telephone	Changed at	Created at	Mobile (MO)	+91-9345678123	Apr 24, 2019, 2:48:21 PM	Apr 24, 2019, 2:59:37 PM >
Category	Telephone	Changed at	Created at								
Mobile (MO)	+91-9345678123	Apr 24, 2019, 2:48:21 PM	Apr 24, 2019, 2:59:37 PM >								
Edit Delete											

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

[<< Frontend Consumption](#)

[<< Top](#)