

[Ask a Question](#) [Write a Blog Post](#)[Login](#)

James Wood

January 16, 2013 | More than 30 minute read

# Navigating the BOPF: Part 3 – Working with the BOPF API

[Follow](#)

65 45 119,015

Like

RSS Feed

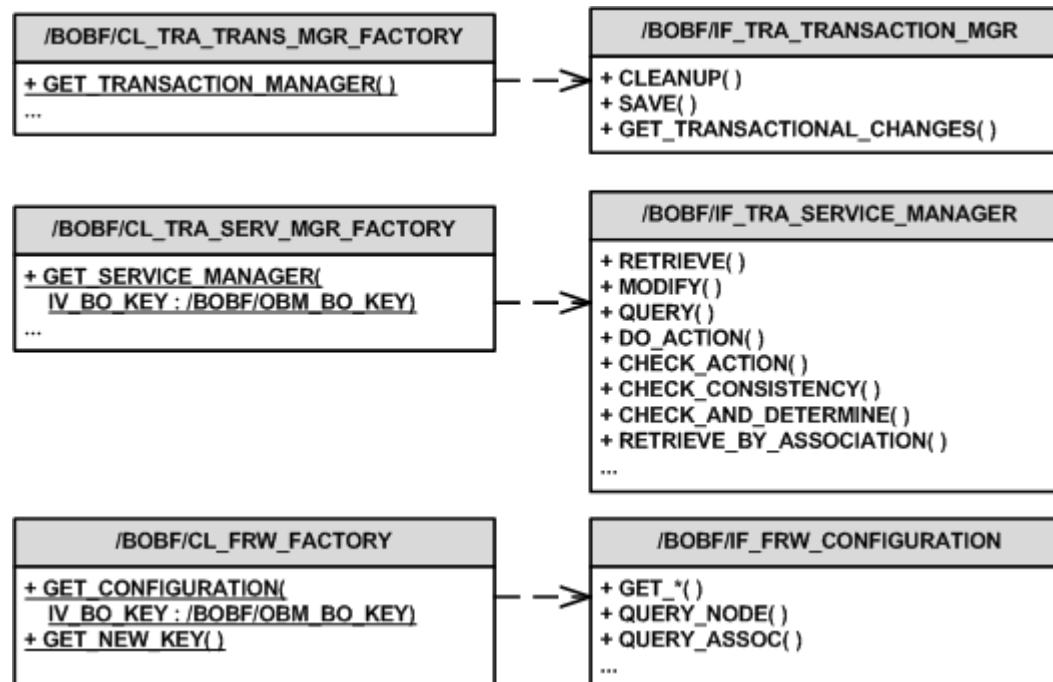
In my previous blog [post](#), we explored the anatomy of business objects within the BOPF. There, we were able to observe the various entities that make up a BO: nodes/attributes, actions, associations, determinations, validations, and queries. Now that you have a feel for what these entities are, we're ready to begin taking a look at the API that is used to manipulate these entities. To guide us through this demonstration, we'll explore the construction of a simple ABAP report program used to perform CRUD operations on a sample BOPF BO shipped by default by SAP: /BOBF/DEMO\_CUSTOMER. You can download the complete example program source code [here](#).

**Note:** The code bundle described above has been enhanced as of 9/18/2013. The code was reworked to factor out a BOPF utilities class of sorts and also demonstrate how to traverse over to dependent objects (DOs).

## BOPF API Overview

Before we begin coding with the BOPF API, let's first take a look at its basic structure. The UML class diagram below highlights some of the main classes that make up the BOPF API. At the end of the day, there are three main objects that we'll be working with to perform most of the operations within the BOPF:

- **/BOBF/IF\_TRA\_TRANSACTION\_MGR**
  - This object reference provides a transaction manager which can be used to manage transactional changes. Such transactions could contain a single step (e.g. update node X) or be strung out across multiple steps (add a node, call an action, and so on).
- **/BOBF/IF\_TRA\_SERVICE\_MANAGER**
  - The service manager object reference provides us with the methods we need to lookup BO nodes, update BO nodes, trigger validations, perform actions, and so on.
- **/BOBF/IF\_FRW\_CONFIGURATION**
  - This object reference provides us with metadata for a particular BO. We'll explore the utility of having access to this metadata coming up shortly.

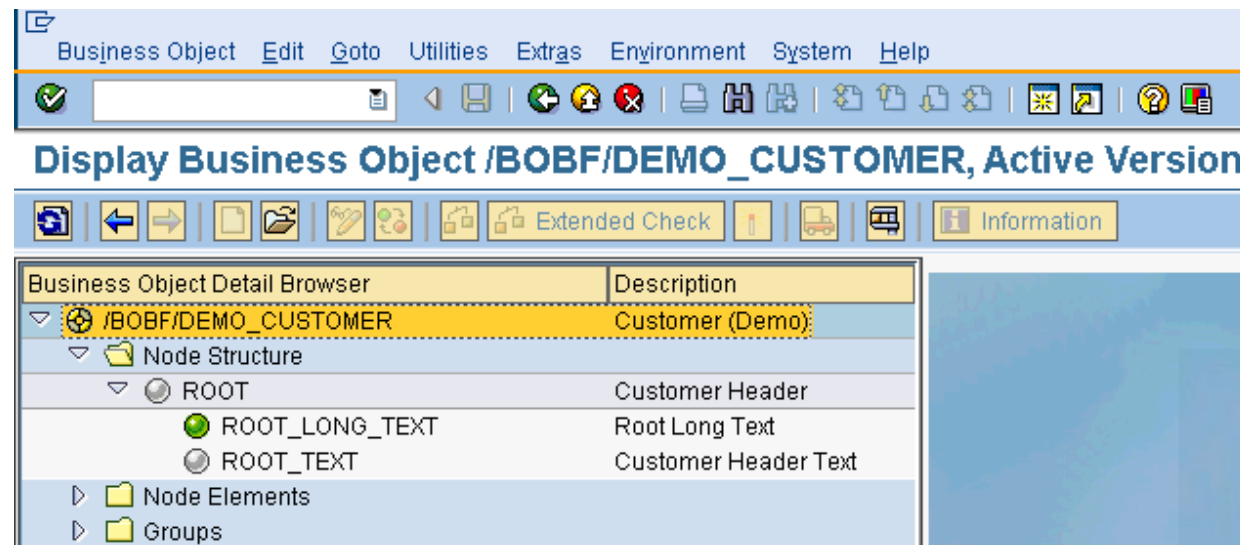


In the upcoming sections, I'll show you how these various API classes collaborate in typical BOPF use cases. Along the way, we'll encounter other useful classes that can be used to perform specific tasks. You can find a complete class listing within package /BOBF/MAIN.

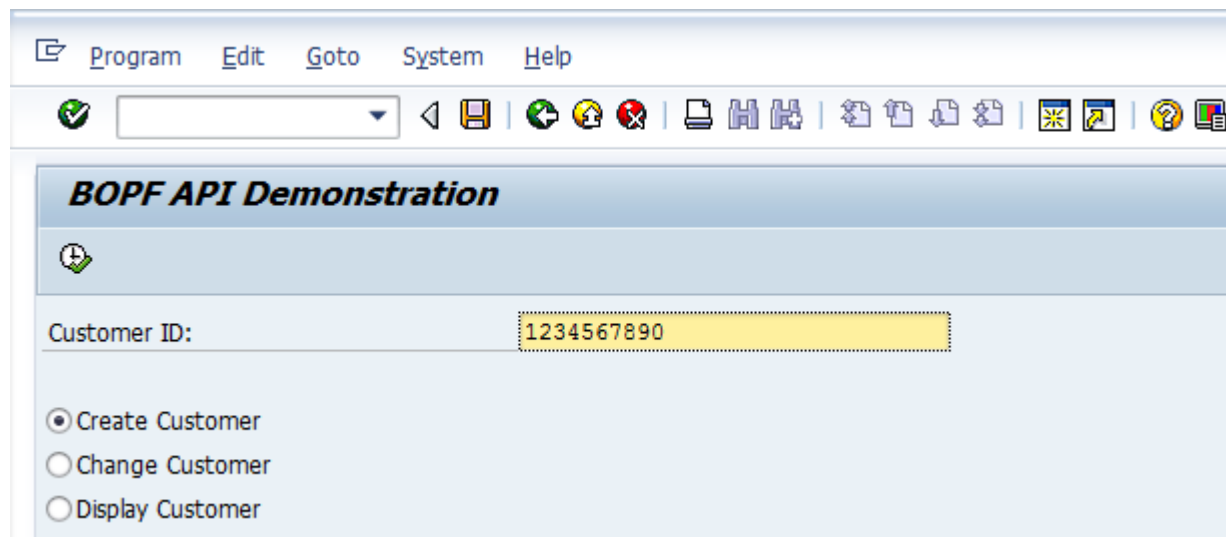
Note: As you'll soon see, the BOPF API is extremely generic in nature. While this provides tremendous flexibility, it also adds a certain amount of tedium to common tasks. Thus, in many applications, you may find that SAP has elected to wrap the API up in another API that is more convenient to work with. For example, in the SAP EHSM solution, SAP defines an "Easy Node Access" API which simplifies the way that developers traverse BO nodes, perform updates, and so on.

## Case Study: Building a Simple Report Program to Manipulate Customer Objects

To demonstrate the BOPF API, we'll build a custom report program which performs basic CRUD operations on a sample BO provided by SAP: /BOBF/DEMO\_CUSTOMER. The figure below shows the makeup of this BO in Transaction /BOBF/CONF\_UI.



Our sample program provides a basic UI as shown below. Here, users have the option of creating, changing, and displaying a particular customer using its ID number. Sort of a simplified Transaction XK01-XK03 if you will.



## Getting Started

To drive the application functionality, we'll create a local test driver class called LCL\_DEMO. As you can see in the code excerpt below, this test driver class loads the core BOPF API objects at setup whenever the CONSTRUCTOR method is invoked. Here, the factory classes illustrated in the UML class diagram shown in the previous section are used to load the various object references.

```
CLASS lcl_demo DEFINITION CREATE PRIVATE.
  PRIVATE SECTION.
    DATA mo_txn_mgr TYPE REF TO /bobf/if_tra_transaction_mgr.
    DATA mo_svc_mgr TYPE REF TO /bobf/if_tra_service_manager.
    DATA mo_bo_conf TYPE REF TO /bobf/if_frw_configuration.

    METHODS:
      constructor RAISING /bobf/cx_frw.
ENDCLASS.
```

```
CLASS lcl_demo IMPLEMENTATION.
  METHOD constructor.
    "Obtain a reference to the BOPF transaction manager:
```

```
me->mo_txn_mgr =  
    /bobf/cl_tra_trans_mgr_factory=>get_transaction_manager( ).  
  
"Obtain a reference to the BOPF service manager:  
me->mo_svc_mgr =  
    /bobf/cl_tra_serv_mgr_factory=>get_service_manager(  
        /bobf/if_demo_customer_c=>sc_bo_key ).  
  
"Access the metadata for the /BOBF/DEMO_CUSTOMER BO:  
me->mo_bo_conf =  
    /bobf/cl_frw_factory=>get_configuration(  
        /bobf/if_demo_customer_c=>sc_bo_key ).  
ENDMETHOD.                " METHOD constructor  
ENDCLASS.
```

For the most part, this should seem fairly straightforward. However, you might be wondering where I came up with the `IV_BO_KEY` parameter in the `GET_SERVICE_MANAGER()` and `GET_CONFIGURATION()` factory method calls. This value is provided to us via the BO's constants interface (`/BOBF/IF_DEMO_CUSTOMER_C` in this case) which can be found within the BO configuration in Transaction `/BOBF/CONF_UI` (see below). This auto-generated constants interface provides us with a convenient mechanism for addressing a BO's key, its defined nodes, associations, queries, and so on. We'll end up using this interface quite a bit during the course of our development.

The screenshot displays the SAP Business Object Detail Browser for the object `/BOBF/DEMO_CUSTOMER`. The interface includes a menu bar (Business Object, Edit, Goto, Utilities, Extras, Environment, System, Help) and a toolbar with various icons. The main window is titled "Display Business Object /BOBF/DEMO\_CUSTOMER, Active Version".

On the left, the "Business Object Detail Browser" shows a tree structure under `/BOBF/DEMO_CUSTOMER`:

- Node Structure
  - ROOT
    - ROOT\_LONG\_TEXT
    - ROOT\_TEXT
  - Node Elements
  - Groups

On the right, the "Business Object" tab is active, showing the following fields:

- Business Object: `/BOBF/DEMO_CUSTOMER`
- Description: Customer (Demo)
- Super Bus. Object:
- Object Category: Master Data Object
- Namespace:
- Prefix:
- ☐ Business Object Model generated
- ☐ Business Object is final
- ☐ Business Object can be enhanced
- ☐ Business Object is abstract
- ☐ Business Object is GenIL enabled


Below these fields is the "Business Object Settings" section:

















- Root Node: ROOT
- Buffer Class: `/BOBF/CL BUF DISPATCHER`
- Constants Interface: `/BOBF/IF DEMO CUSTOMER C` (circled in red)
- Status Class:
- Status Dispatcher:

A red arrow points from the `ROOT` node in the tree to the `/BOBF/IF DEMO CUSTOMER C` field in the Settings section.

Interface /BOBF/IF\_DEMO\_CUSTOMER\_C Implemented / Active

Properties Interfaces **Attributes** Methods Events Types Aliases

 ☐ Filter

Attribute	Level	Re...	Typing	Associated...		Description	Initial value
SC_ACTION	Consta..	<input type="checkbox"/>				Actions	
SC_ACTION_ATTRIBUTE	Consta..	<input type="checkbox"/>				Action Parameter Attributes	
SC_ALTERNATIVE_KEY	Consta..	<input type="checkbox"/>				Alternative Keys	
SC_ASSOCIATION	Consta..	<input type="checkbox"/>				Associations	
SC_ASSOCIATION_ATTRIBUTE	Consta..	<input type="checkbox"/>				Association Parameter Attributes	
SC_BO_KEY	Consta..	<input type="checkbox"/>				Customer (Demo)	
SC_BO_NAME	Consta..	<input type="checkbox"/>				Customer (Demo)	
SC_BO_PROXY_NAME	Consta..	<input type="checkbox"/>				Customer (Demo)	
SC_DETERMINATION	Consta..	<input type="checkbox"/>				Determinations	
SC_NODE	Consta..	<input type="checkbox"/>				Nodes	
SC_NODE_ATTRIBUTE	Consta..	<input type="checkbox"/>				Node Attributes	
SC_NODE_CATEGORY	Consta..	<input type="checkbox"/>				Node Categories	
SC_QUERY	Consta..	<input type="checkbox"/>				Queries	
SC_QUERY_ATTRIBUTE	Consta..	<input type="checkbox"/>				Query Attributes	
SC_VALIDATION	Consta..	<input type="checkbox"/>				Validations	
		<input type="checkbox"/>	Type				

## Creating New Customers

Once we have the basic framework in place, we are ready to commence with the development of the various CRUD operations that our application will support. To get things started, we'll take a look at the creation of a new customer instance. For the most part, this involves little more than a call to the `MODIFY()` method of the `/BOBF/IF_TRA_SERVICE_MANAGER` object reference. Of course, as you can see in the code excerpt below, there is a fair amount of setup that we must do before we can call this method.

```
CLASS lcl_demo DEFINITION CREATE PRIVATE.
  PUBLIC SECTION.
    CLASS-METHODS:
      create_customer IMPORTING iv_customer_id
                     TYPE /bobf/demo_customer_id.
  ...
ENDCLASS.
```

CLASS lcl\_demo IMPLEMENTATION.

METHOD create\_customer.

"Method-Local Data Declarations:

DATA lo\_driver TYPE REF TO lcl\_demo.

DATA lt\_mod TYPE /bobf/t\_frw\_modification.

DATA lo\_change TYPE REF TO /bobf/if\_tra\_change.

DATA lo\_message TYPE REF TO /bobf/if\_frw\_message.

DATA lv\_rejected TYPE boole\_d.

DATA lx\_bopf\_ex TYPE REF TO /bobf/cx\_frw.

DATA lv\_err\_msg TYPE string.

DATA lr\_s\_root TYPE REF TO /bobf/s\_demo\_customer\_hdr\_k.

DATA lr\_s\_txt TYPE REF TO /bobf/s\_demo\_short\_text\_k.

DATA lr\_s\_txt\_hdr TYPE REF TO /bobf/s\_demo\_longtext\_hdr\_k.

DATA lr\_s\_txt\_cont TYPE REF TO /bobf/s\_demo\_longtext\_item\_k.

FIELD-SYMBOLS:

<ls\_mod> LIKE LINE OF lt\_mod.

"Use the BOPF API to create a new customer record:

TRY.

"Instantiate the driver class:

CREATE OBJECT lo\_driver.

"Build the ROOT node:

CREATE DATA lr\_s\_root.

lr\_s\_root->key = /bobf/cl\_frw\_factory=>get\_new\_key( ).



```
lr_s_root->customer_id    = iv_customer_id.  
lr_s_root->sales_org       = 'AMER'.  
lr_s_root->cust_curr       = 'USD'.  
lr_s_root->address_contry  = 'US'.  
lr_s_root->address         = '1234 Any Street'.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.  
<ls_mod>-node             = /bobf/if_demo_customer_c=>sc_node-root.  
<ls_mod>-change_mode      = /bobf/if_frw_c=>sc_modify_create.  
<ls_mod>-key              = lr_s_root->key.  
<ls_mod>-data             = lr_s_root.
```

"Build the ROOT\_TEXT node:

```
CREATE DATA lr_s_txt.  
lr_s_txt->key             = /bobf/cl_frw_factory=>get_new_key( ).  
lr_s_txt->text            = 'Sample Customer Record'.  
lr_s_txt->language        = sy-langu.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.  
<ls_mod>-node             = /bobf/if_demo_customer_c=>sc_node-root_text.  
<ls_mod>-change_mode      = /bobf/if_frw_c=>sc_modify_create.  
<ls_mod>-source_node      = /bobf/if_demo_customer_c=>sc_node-root.  
<ls_mod>-association      =
```

```
    /bobf/if_demo_customer_c=>sc_association-root-root_text.  
<ls_mod>-source_key       = lr_s_root->key.  
<ls_mod>-key              = lr_s_txt->key.  
<ls_mod>-data             = lr_s_txt.
```

"Build the ROOT\_LONG\_TEXT node:

"If you look at the node type for this node, you'll notice that  
"it's a "Delegated Node". In other words, it is defined in terms  
"of the /BOBF/DEMO\_TEXT\_COLLECTION business object. The following  
"code accounts for this indirection.

```
CREATE DATA lr_s_txt_hdr.
```

```
lr_s_txt_hdr->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node = /bobf/if_demo_customer_c=>sc_node-root_long_text.
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node = /bobf/if_demo_customer_c=>sc_node-root.
```

```
<ls_mod>-association =
```

```
    /bobf/if_demo_customer_c=>sc_association-root-root_long_text.
```

```
<ls_mod>-source_key = lr_s_root->key.
```

```
<ls_mod>-key = lr_s_txt_hdr->key.
```

```
<ls_mod>-data = lr_s_txt_hdr.
```

"Create the CONTENT node:

```
CREATE DATA lr_s_txt_cont.
```

```
lr_s_txt_cont->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_txt_cont->language = sy-langu.
```

```
lr_s_txt_cont->text_type = 'MEMO'.
```

```
lr_s_txt_cont->text_content = 'Demo customer created via BOPF API.'.
```

```

APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
<ls_mod>-node          =

    lo_driver->mo_bo_conf->query_node(

        iv_proxy_node_name = 'ROOT_LONG_TXT.CONTENT' ).
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
<ls_mod>-source_node = /bobf/if_demo_customer_c=>sc_node-root_long_text.
<ls_mod>-source_key  = lr_s_txt_hdr->key.
<ls_mod>-key         = lr_s_txt_cont->key.
<ls_mod>-data        = lr_s_txt_cont.

<ls_mod>-association =
    lo_driver->mo_bo_conf->query_assoc(
        iv_node_key   = /bobf/if_demo_customer_c=>sc_node-root_long_text
        iv_assoc_name = 'CONTENT' ).

"Create the customer record:
CALL METHOD lo_driver->mo_svc_mgr->modify
    EXPORTING
        it_modification = lt_mod
    IMPORTING
        eo_change       = lo_change
        eo_message      = lo_message.

"Check for errors:
IF lo_message IS BOUND.
    IF lo_message->check( ) EQ abap_true.
        lo_driver->display_messages( lo_message ).
    RETURN.

```

```
ENDIF.  
ENDIF.  
  
"Apply the transactional changes:  
CALL METHOD lo_driver->mo_txn_mngr->save  
IMPORTING  
    eo_message = lo_message  
    ev_rejected = lv_rejected.  
  
IF lv_rejected EQ abap_true.  
    lo_driver->display_messages( lo_message ).  
    RETURN.  
ENDIF.  
  
"If we get to here, then the operation was successful:  
WRITE: / 'Customer', iv_customer_id, 'created successfully.'.  
CATCH /bobf/cx_frw INTO lx_bopf_ex.  
    lv_err_msg = lx_bopf_ex->get_text( ).  
    WRITE: / lv_err_msg.  
ENDTRY.  
ENDMETHOD.          " METHOD create_customer  
ENDCLASS.
```

As you can see in the code excerpt above, the majority of the code is devoted to building a table which is passed in the `IT_MODIFICATION` parameter of the `MODIFY()` method. Here, a separate record is created for each node row that is being modified (or inserted in this case). This record contains information such as the node object key (`NODE`), the edit mode (`CHANGE_MODE`), the row key (`KEY`) which is an auto-generated GUID, association/parent key information, and of course, the actual data (`DATA`). If you've ever worked with ALE IDocs, then this will probably feel vaguely familiar.

Looking more closely at the population of the node row data, you can see that we're working with data references which are created dynamically using the CREATE DATA statement. This indirection is necessary since the BOPF API is generic in nature. You can find the structure definitions for each node by double-clicking on the node in Transaction /BOBF/CONF\_UI and looking at the *Combined Structure* field (see below).

Once the modification table is filled out, we can call the MODIFY() method to insert the record(s). Assuming all is successful, we can then commit the transaction by calling the SAVE() method on the /BOBF/IF\_TRA\_TRANSACTION\_MANAGER instance. Should any errors occur, we can display the error messages using methods of the /BOBF/IF\_FRW\_MESSAGE object reference which is returned from both methods. This is evidenced by the simple utility method DISPLAY\_MESSAGES() shown below. That's pretty much all there is to it.

```
CLASS lcl_demo DEFINITION CREATE PRIVATE.  
  PRIVATE SECTION.  
    METHODS:  
      display_messages IMPORTING io_message  
                       TYPE REF TO /bobf/if_frw_message.  
ENDCLASS.
```

```
CLASS lcl_demo IMPLEMENTATION.  
  METHOD display_messages.  
    "Method-Local Data Declarations:  
    DATA lt_messages TYPE /bobf/t_frw_message_k.  
    DATA lv_msg_text TYPE string.  
    FIELD-SYMBOLS <ls_message> LIKE LINE OF lt_messages.  
  
    "Sanity check:  
    CHECK io_message IS BOUND.
```

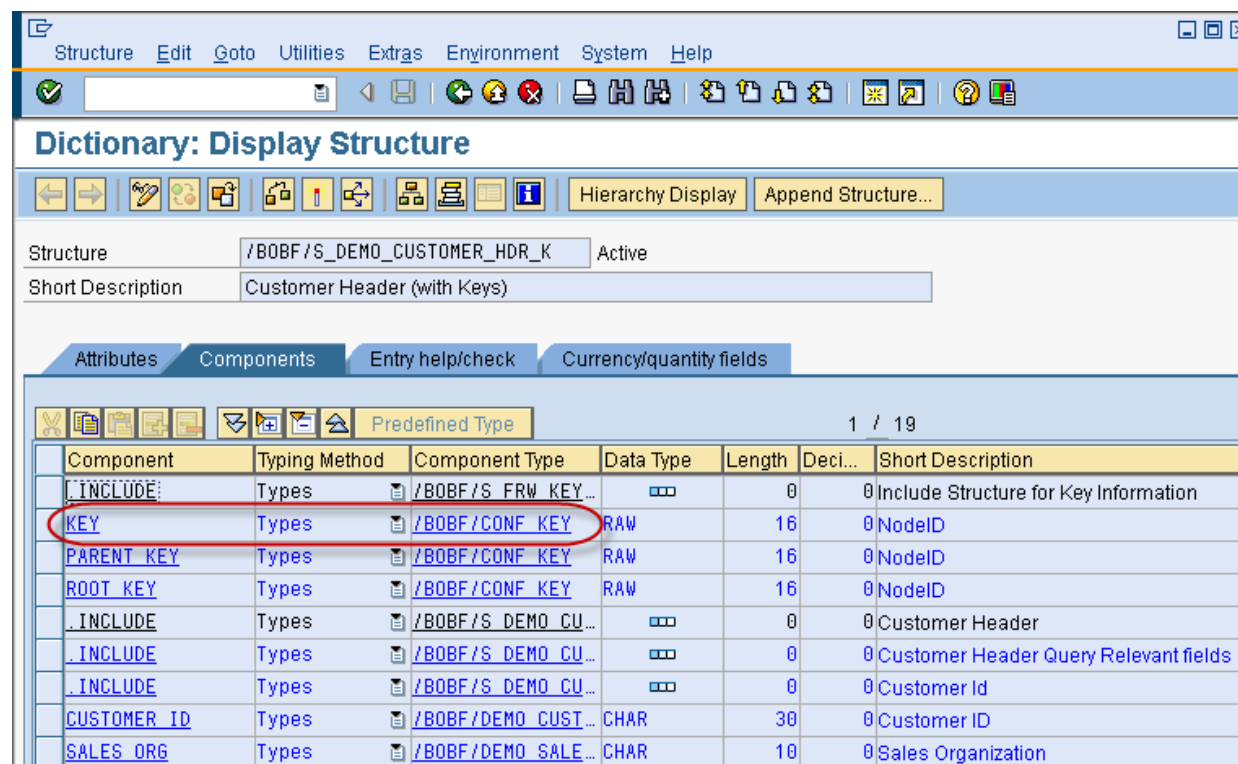
```

"Output each of the messages in the collection:
io_message->get_messages( IMPORTING et_message = lt_messages ).
LOOP AT lt_messages ASSIGNING <ls_message>.
    lv_msg_text = <ls_message>-message->get_text( ).
    WRITE: / lv_msg_text.
ENDLOOP.
ENDMETHOD.                " METHOD display_messages
ENDCLASS.

```

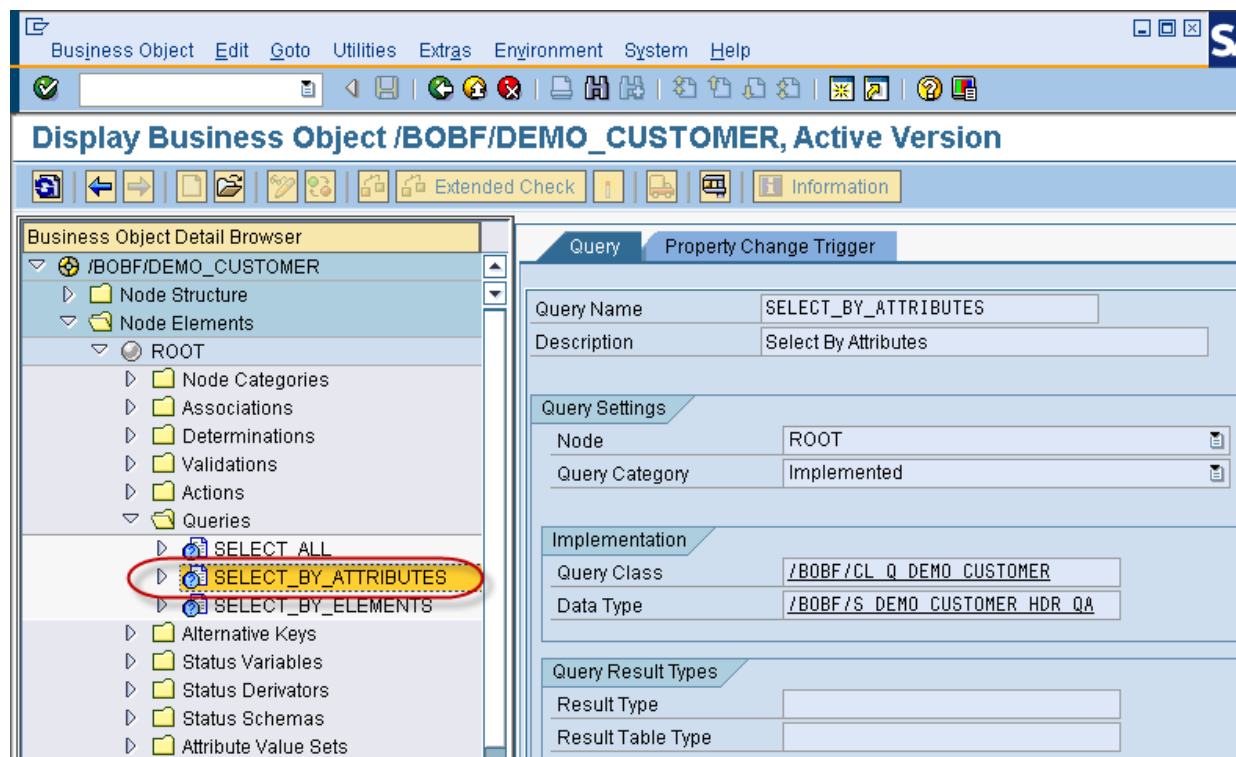
## Performing Customer Queries

If you look closely at the customer creation code illustrated in the previous section, you can see that each node row is keyed by an auto-generated GUID of type /BOBF/CONF\_KEY (see below). Since most users don't happen to have 32-character hex strings memorized, we typically have to resort to queries if we want to find particular BO instances. For example, in our customer demo program, we want to provide a way for users to lookup customers using their customer ID value. Of course, we could have just as easily defined an alternative query selection to pull the customer records.



Component	Typing Method	Component Type	Data Type	Length	Deci...	Short Description
[INCLUDE]	Types	/BOBF/S_FRW_KEY...		0		0 Include Structure for Key Information
<b>KEY</b>	Types	/BOBF/CONF_KEY	RAW	16	0	0 NodeID
PARENT_KEY	Types	/BOBF/CONF_KEY	RAW	16	0	0 NodeID
ROOT_KEY	Types	/BOBF/CONF_KEY	RAW	16	0	0 NodeID
.INCLUDE	Types	/BOBF/S_DEMO CU...		0		0 Customer Header
.INCLUDE	Types	/BOBF/S_DEMO CU...		0		0 Customer Header Query Relevant fields
.INCLUDE	Types	/BOBF/S_DEMO CU...		0		0 Customer Id
CUSTOMER_ID	Types	/BOBF/DEMO CUST...	CHAR	30		0 Customer ID
SALES_ORG	Types	/BOBF/DEMO SALE...	CHAR	10		0 Sales Organization

As we learned in the previous blog post, most BOs come with one or more queries which allow us to search for BOs according to various node criteria. In the case of the /BOBF/DEMO\_CUSTOMER business object, we want to use the SELECT\_BY\_ATTRIBUTES query attached to the ROOT node (see below). This allows us to lookup customers by their ID value.



The code excerpt below shows how we defined our query in a method called GET\_CUSTOMER\_FOR\_ID(). As you can see, the query is executed by calling the aptly named QUERY() method of the /BOBF/IF\_TRA\_SERVICE\_MANAGER instance. The query parameters are provided in the form of an internal table of type /BOBF/T\_FRW\_QUERY\_SELPARAM. This table type has a similar look and feel to a range table or SELECT-OPTION. The results of the query are returned in a table of type /BOBF/T\_FRW\_KEY. This table contains the keys of the node rows that matched the query parameters. In our sample case, there should be only one match, so we simply return the first key in the list.

```
CLASS lcl_demo DEFINITION CREATE PRIVATE.
  PRIVATE SECTION.
    METHODS:
      get_customer_for_id IMPORTING iv_customer_id
```

```

        TYPE /bobf/demo_customer_id
    RETURNING VALUE(rv_customer_key)
        TYPE /bobf/conf_key
    RAISING /bobf/cx_frw.

ENDCLASS.

```

```

CLASS lcl_demo IMPLEMENTATION.

```

```

    METHOD get_customer_for_id.

```

```

        "Method-Local Data Declarations:

```

```

        DATA lo_driver      TYPE REF TO lcl_demo.

```

```

        DATA lt_parameters  TYPE /bobf/t_frw_query_selparam.

```

```

        DATA lt_customer_keys TYPE /bobf/t_frw_key.

```

```

        DATA lx_bopf_ex     TYPE REF TO /bobf/cx_frw.

```

```

        DATA lv_err_msg     TYPE string.

```

```

        FIELD-SYMBOLS <ls_parameter> LIKE LINE OF lt_parameters.

```

```

        FIELD-SYMBOLS <ls_customer_key> LIKE LINE OF lt_customer_keys.

```

```

        "Instantiate the test driver class:

```

```

        CREATE OBJECT lo_driver.

```

```

        "Though we could conceivably lookup the customer using an SQL query,

```

```

        "the preferred method of selection is a BOPF query:

```

```

        APPEND INITIAL LINE TO lt_parameters ASSIGNING <ls_parameter>.

```

```

        <ls_parameter>-attribute_name =

```

```

            /bobf/if_demo_customer_c=>sc_query_attribute-root-select_by_attributes-customer_id.

```

```

        <ls_parameter>-sign          = 'I'.

```



```

<ls_parameter>-option      = 'EQ'.
<ls_parameter>-low         = iv_customer_id.

CALL METHOD lo_driver->mo_svc_mgr->query
EXPORTING
    iv_query_key            =

    /bobf/if_demo_customer_c=>sc_query-root-select_by_attributes
    it_selection_parameters = lt_parameters
IMPORTING
    et_key                  = lt_customer_keys.

"Return the matching customer's KEY value:
READ TABLE lt_customer_keys INDEX 1 ASSIGNING <ls_customer_key>.
IF sy-subrc EQ 0.
    rv_customer_key = <ls_customer_key>-key.
ENDIF.
ENDMETHOD.          " METHOD get_customer_for_id
ENDCLASS.

```

## Displaying Customer Records

With the query logic now in place, we now know which customer record to lookup. The question is, how do we retrieve it? For this task, we must use the

RETRIEVE() and RETRIEVE\_BY\_ASSOCIATION() methods provided by the /BOBF/IF\_TRA\_SERVICE\_MANAGER instance. As simple as this sounds, the devil is in the details. Here, in addition to constructing the calls to the RETRIEVE\*() methods, we must also dynamically define the result tables which will be used to store the results.

As you can see in the code excerpt below, we begin our search by accessing the customer ROOT node using the RETRIEVE() method. Here, the heavy lifting is performed by the GET\_NODE\_ROW() and GET\_NODE\_TABLE() helper methods. Looking at the implementation of the GET\_NODE\_TABLE() method, you can see how we're using the

/BOBF/IF\_FRW\_CONFIGURATION object reference to lookup the node's metadata. This metadata provides us with the information we need to construct an internal table to house the results returned from the RETRIEVE() method. The GET\_NODE\_ROW() method then dynamically retrieves the record located at the index defined by the IV\_INDEX parameter.

Within the DISPLAY\_CUSTOMER() method, we get our hands on the results by performing a cast on the returned structure reference. From here, we can access the row attributes as per usual.

After the root node has been retrieved, we can traverse to the child nodes of the /BOBF/DEMO\_CUSTOMER object using the RETRIEVE\_BY\_ASSOCIATION() method. Here, the process is basically the same. The primary difference is in the way we lookup the association metadata which is used to build the call to RETRIEVE\_BY\_ASSOCIATION(). Once again, we perform a cast on the returned structure reference and display the sub-node attributes from there.

```

CLASS lcl_demo DEFINITION CREATE PRIVATE.
  PUBLIC SECTION.
    CLASS-METHODS:
      display_customer IMPORTING iv_customer_id
                      TYPE /bobf/demo_customer_id.

  PRIVATE SECTION.

    METHODS:
      get_node_table IMPORTING iv_key TYPE /bobf/conf_key
                    iv_node_key TYPE /bobf/obm_node_key
                    iv_edit_mode TYPE /bobf/conf_edit_mode

                    DEFAULT /bobf/if_conf_c=>sc_edit_read_only
                    RETURNING VALUE(rr_data) TYPE REF TO data
                    RAISING /bobf/cx_frw,

      get_node_row IMPORTING iv_key TYPE /bobf/conf_key
                    iv_node_key TYPE /bobf/obm_node_key

```

```

        iv_edit_mode TYPE /bobf/conf_edit_mode

        DEFAULT /bobf/if_conf_c=>sc_edit_read_only
        iv_index TYPE i DEFAULT 1
RETURNING VALUE(rr_data) TYPE REF TO data
        RAISING /bobf/cx_frw,

get_node_table_by_assoc IMPORTING iv_key TYPE /bobf/conf_key
                        iv_node_key TYPE /bobf/obm_node_key
                        iv_assoc_key TYPE /bobf/obm_assoc_key
                        iv_edit_mode TYPE /bobf/conf_edit_mode

                        DEFAULT /bobf/if_conf_c=>sc_edit_read_only
RETURNING VALUE(rr_data) TYPE REF TO data
        RAISING /bobf/cx_frw,

get_node_row_by_assoc IMPORTING iv_key TYPE /bobf/conf_key
                        iv_node_key TYPE /bobf/obm_node_key
                        iv_assoc_key TYPE /bobf/obm_assoc_key
                        iv_edit_mode TYPE /bobf/conf_edit_mode

                        DEFAULT /bobf/if_conf_c=>sc_edit_read_only
                        iv_index TYPE i DEFAULT 1
RETURNING VALUE(rr_data) TYPE REF TO data
        RAISING /bobf/cx_frw.

ENDCLASS.

CLASS lcl_demo IMPLEMENTATION.
    METHOD display_customer.

```

"Method-Local Data Declarations:

DATA lo\_driver TYPE REF TO lcl\_demo.

DATA lv\_customer\_key TYPE /bobf/conf\_key.

DATA lx\_bopf\_ex TYPE REF TO /bobf/cx\_frw.

DATA lv\_err\_msg TYPE string.

DATA lr\_s\_root TYPE REF TO /bobf/s\_demo\_customer\_hdr\_k.

DATA lr\_s\_text TYPE REF TO /bobf/s\_demo\_short\_text\_k.

"Try to display the selected customer:

TRY.

"Instantiate the test driver class:

CREATE OBJECT lo\_driver.

"Lookup the customer's key attribute using a query:

lv\_customer\_key = lo\_driver->get\_customer\_for\_id( iv\_customer\_id ).

"Display the header-level details for the customer:

lr\_s\_root ?=

lo\_driver->get\_node\_row(

iv\_key = lv\_customer\_key

iv\_node\_key = /bobf/if\_demo\_customer\_c=>sc\_node-root

iv\_index = 1 ).

WRITE: / 'Display Customer', lr\_s\_root->customer\_id.

ULINE.

```
WRITE: / 'Sales Organization:', lr_s_root->sales_org.
WRITE: / 'Address:', lr_s_root->address.
SKIP.
```

"Traverse to the ROOT\_TEXT node to display the customer short text:

```
lr_s_text ?=
  lo_driver->get_node_row_by_assoc(

    iv_key = lv_customer_key
    iv_node_key = /bobf/if_demo_customer_c=>sc_node-root
    iv_assoc_key = /bobf/if_demo_customer_c=>sc_association-root-root_text
    iv_index = 1 ).
WRITE: / 'Short Text:', lr_s_text->text.
CATCH /bobf/cx_frw INTO lx_bopf_ex.
  lv_err_msg = lx_bopf_ex->get_text( ).
WRITE: / lv_err_msg.
ENDTRY.
ENDMETHOD.          " METHOD display_customer
```

METHOD get\_node\_table.

"Method-Local Data Declarations:

```
DATA lt_key          TYPE /bobf/t_frw_key.
DATA ls_node_conf    TYPE /bobf/s_confro_node.
DATA lo_change       TYPE REF TO /bobf/if_tra_change.

DATA lo_message      TYPE REF TO /bobf/if_frw_message.
```

```
FIELD-SYMBOLS <ls_key> LIKE LINE OF lt_key.
FIELD-SYMBOLS <lt_data> TYPE INDEX TABLE.
```

"Lookup the node's configuration:

```
CALL METHOD mo_bo_conf->get_node
EXPORTING
  iv_node_key = iv_node_key
IMPORTING
  es_node     = ls_node_conf.
```

"Use the node configuration metadata to create the result table:

```
CREATE DATA rr_data TYPE (ls_node_conf-data_table_type).
ASSIGN rr_data->* TO <lt_data>.
```

"Retrieve the target node:

```
APPEND INITIAL LINE TO lt_key ASSIGNING <ls_key>.
<ls_key>-key = iv_key.
```

```
CALL METHOD mo_svc_mgr->retrieve
```

```
EXPORTING
  iv_node_key = iv_node_key
  it_key      = lt_key
IMPORTING
  eo_message  = lo_message
  eo_change   = lo_change
  et_data     = <lt_data>.
```

```
"Check the results:
IF lo_message IS BOUND.
  IF lo_message->check( ) EQ abap_true.
    display_messages( lo_message ).
    RAISE EXCEPTION TYPE /bobf/cx_dac.
  ENDIF.
ENDIF.
ENDMETHOD.                " METHOD get_node_table
```

```
METHOD get_node_row.
"Method-Local Data Declarations:
DATA lr_t_data TYPE REF TO data.

FIELD-SYMBOLS <lt_data> TYPE INDEX TABLE.
FIELD-SYMBOLS <ls_row> TYPE ANY.
```

```
"Lookup the node data:
lr_t_data =
  get_node_table( iv_key      = iv_key
                  iv_node_key = iv_node_key
                  iv_edit_mode = iv_edit_mode ).
```

```
IF lr_t_data IS NOT BOUND.
  RAISE EXCEPTION TYPE /bobf/cx_dac.
ENDIF.
```

```

"Try to pull the record at the specified index:
ASSIGN lr_t_data->* TO <lt_data>.
READ TABLE <lt_data> INDEX iv_index ASSIGNING <ls_row>.
IF sy-subrc EQ 0.
    GET REFERENCE OF <ls_row> INTO rr_data.
ELSE.
    RAISE EXCEPTION TYPE /bobf/cx_dac.
ENDIF.
ENDMETHOD.                " METHOD get_node_row

```

```

METHOD get_node_table_by_assoc.
"Method-Local Data Declarations:
DATA lt_key          TYPE /bobf/t_frw_key.
DATA ls_node_conf    TYPE /bobf/s_confro_node.
DATA ls_association TYPE /bobf/s_confro_assoc.
DATA lo_change       TYPE REF TO /bobf/if_tra_change.
DATA lo_message      TYPE REF TO /bobf/if_frw_message.

```

```

FIELD-SYMBOLS <ls_key> LIKE LINE OF lt_key.
FIELD-SYMBOLS <lt_data> TYPE INDEX TABLE.

```

```

"Lookup the association metadata to find out more
"information about the target sub-node:
CALL METHOD mo_bo_conf->get_assoc
EXPORTING
    iv_assoc_key = iv_assoc_key
    iv_node_key  = iv_node_key
IMPORTING
    es_assoc     = ls_association.

```



```
IF ls_association-target_node IS NOT BOUND.  
  RAISE EXCEPTION TYPE /bobf/cx_dac.  
ENDIF.
```

"Use the node configuration metadata to create the result table:

```
ls_node_conf = ls_association-target_node->*.
```

```
CREATE DATA rr_data TYPE (ls_node_conf-data_table_type).  
ASSIGN rr_data->* TO <lt_data>.
```

```
"Retrieve the target node:  
APPEND INITIAL LINE TO lt_key ASSIGNING <ls_key>.  
<ls_key>-key = iv_key.
```

```
CALL METHOD mo_svc_mgr->retrieve_by_association  
  EXPORTING  
    iv_node_key    = iv_node_key  
    it_key         = lt_key  
    iv_association = iv_assoc_key  
    iv_fill_data   = abap_true  
  IMPORTING  
    eo_message     = lo_message  
    eo_change      = lo_change  
    et_data        = <lt_data>.
```

```
"Check the results:
IF lo_message IS BOUND.
  IF lo_message->check( ) EQ abap_true.
    display_messages( lo_message ).
    RAISE EXCEPTION TYPE /bobf/cx_dac.
  ENDIF.
ENDIF.
ENDMETHOD.                " METHOD get_node_table_by_assoc
```

```
METHOD get_node_row_by_assoc.
  "Method-Local Data Declarations:
  DATA lr_t_data TYPE REF TO data.
```

```
FIELD-SYMBOLS <lt_data> TYPE INDEX TABLE.
FIELD-SYMBOLS <ls_row> TYPE ANY.
```

```
"Lookup the node data:
lr_t_data =
  get_node_table_by_assoc( iv_key      = iv_key
                          iv_node_key = iv_node_key
                          iv_assoc_key = iv_assoc_key
                          iv_edit_mode = iv_edit_mode ).
```

```
IF lr_t_data IS NOT BOUND.
  RAISE EXCEPTION TYPE /bobf/cx_dac.
```

```

ENDIF.

"Try to pull the record at the specified index:
ASSIGN lr_t_data->* TO <lt_data>.
READ TABLE <lt_data> INDEX iv_index ASSIGNING <ls_row>.
IF sy-subrc EQ 0.
    GET REFERENCE OF <ls_row> INTO rr_data.
ELSE.
    RAISE EXCEPTION TYPE /bobf/cx_dac.
ENDIF.
ENDMETHOD.          " METHOD get_node_row_by_assoc
ENDCLASS.

```

Note: In this simple example, we didn't bother to drill down to display the contents of the ROOT\_LONG\_TEXT node. However, if we had wanted to do so, we would have needed to create a separate service manager instance for the /BOBF/DEMO\_TEXT\_COLLECTION business object since the data within that node is defined by that delegated BO as opposed to the /BOBF/DEMO\_CUSTOMER BO. Otherwise, the process is the same.

## Modifying Customer Records

The process of modifying a customer record essentially combines logic from the display and create functions. The basic process is as follows:

1. First, we perform a query to find the target customer record.
2. Next, we use the RETRIEVE\*( ) methods to retrieve the node rows we wish to modify. Using the returned structure references, we can modify the target attributes using simple assignment statements.
3. Finally, we collect the node row changes into the modification table that is fed into MODIFY( ) method provided by the /BOBF/IF\_TRA\_SERVICE\_MANAGER instance.

The code excerpt below shows how the changes are carried out. Here, we're simply updating the address string on the customer. Of course, we could have performed wholesale changes if we had wanted to.

```
CLASS lcl_demo DEFINITION CREATE PRIVATE.  
  PUBLIC SECTION.  
    CLASS-METHODS:  
      change_customer IMPORTING iv_customer_id  
                      TYPE /bobf/demo_customer_id.  
ENDCLASS.  
  
CLASS lcl_demo IMPLEMENTATION.  
  METHOD change_customer.  
    "Method-Local Data Declarations:  
    DATA lo_driver      TYPE REF TO lcl_demo.  
    DATA lv_customer_key TYPE /bobf/conf_key.  
    DATA lt_mod         TYPE /bobf/t_frw_modification.  
    DATA lo_change      TYPE REF TO /bobf/if_tra_change.  
    DATA lo_message     TYPE REF TO /bobf/if_frw_message.  
    DATA lv_rejected    TYPE boole_d.  
    DATA lx_bopf_ex     TYPE REF TO /bobf/cx_frw.  
    DATA lv_err_msg     TYPE string.  
  
    FIELD-SYMBOLS:  
      <ls_mod> LIKE LINE OF lt_mod.  
  
    DATA lr_s_root TYPE REF TO /bobf/s_demo_customer_hdr_k.  
  
    "Try to change the address on the selected customer:  
    TRY.
```

"Instantiate the test driver class:

```
CREATE OBJECT lo_driver.
```

"Access the customer ROOT node:

```
lv_customer_key = lo_driver->get_customer_for_id( iv_customer_id ).
```

```
lr_s_root ?=
```

```
    lo_driver->get_node_row( iv_key = lv_customer_key  
                           iv_node_key = /bobf/if_demo_customer_c=>sc_node-root  
                           iv_edit_mode = /bobf/if_conf_c=>sc_edit_exclusive  
                           iv_index = 1 ).
```

"Change the address string on the customer:

```
lr_s_root->address = '1234 Boardwalk Ave.'.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node      = /bobf/if_demo_customer_c=>sc_node-root.
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_update.
```

```
<ls_mod>-key       = lr_s_root->key.
```

```
<ls_mod>-data      = lr_s_root.
```

"Update the customer record:

```
CALL METHOD lo_driver->mo_svc_mngr->modify
```

```
    EXPORTING
```

```
        it_modification = lt_mod
```

```
IMPORTING
    eo_change      = lo_change
    eo_message     = lo_message.

"Check for errors:
IF lo_message IS BOUND.
    IF lo_message->check( ) EQ abap_true.
        lo_driver->display_messages( lo_message ).
        RETURN.
    ENDIF.
ENDIF.

"Apply the transactional changes:
CALL METHOD lo_driver->mo_txn_mgr->save
IMPORTING
    eo_message     = lo_message
    ev_rejected    = lv_rejected.

IF lv_rejected EQ abap_true.
    lo_driver->display_messages( lo_message ).
    RETURN.
ENDIF.
```

Alert Moderator

## Assigned Tags

ABAP Development

abap

bopf

```
"If we get to here, then the operation was successful:
WRITE: / 'Customer', iv_customer_id, 'updated successfully.'.
CATCH /bobf/cx_frwr INTO lx_bopf_ex.
lv_err_msg = lx_bopf_ex->get_text( ).
```

floorplan manager

```
WRITE: / lv_err_msg.
```

web dynpro abap

```
ENDTRY.
```

```
ENDMETHOD.
```

```
" METHOD change_customer
```

```
ENDCLASS.
```

## Similar Blog Posts

### [BOPF: Custom Lock/Unlock Action for Legacy DAO](#)

By Gaurav Sharan Jun 29, 2018

I often find that the best way to learn a technology framework is to see how it plays out via code. At this level, we can clearly visualize the relationships between the various entities and see how they perform at runtime. Hopefully after reading this post, you should have a better understanding of how all the BOPF pieces fit together. In my next blog post, we'll expand upon what we've learned and consider some more advanced features of the BOPF API.

### [BOPF-SADL Mapping – Demystifying Limitations](#)

By Ivo Vollrath Jun 06, 2016

### [One truth, multiple views on it: The various BOPF modeling environments](#)

By Oliver Jaegle Sep 03, 2015

## Related Questions

### [Where the integration \( link \) of Bopf\\_ewb and Fpm done](#)

By Former Member Nov 25, 2015

### [How to call another BOPF from action of one BOPF ?](#)

By Surya Sarathi Basu Dec 26, 2018

### [Updating an object from a BADI](#)

By Custodio de Oliveira Jan 15, 2015

## 65 Comments

---

You must be [Logged on](#) to comment or reply to a post.



Aliaksandr Shchurko

May 2, 2013 at 1:13 pm

Thanks.

Like 0 | Share



Former Member

July 4, 2013 at 11:14 am

Very much helpful , thanks !

Like 0 | Share



Former Member

September 17, 2013 at 7:25 pm

Very helpful. Could you please provide me the code to read the long text as well.

Thanks



Like 0 | Share



James Wood | Blog Post Author

September 17, 2013 at 8:21 pm

The complete code bundle can be downloaded from here:

<http://www.bowdark.com/downloads/BOPFDemoProgram.zip>

Thanks.

Like 0 | Share



Former Member

September 18, 2013 at 3:51 pm

James,

I have downloaded the full program. Neither the full program nor the blog post have the logic to read the long text from node ROOT\_LONG\_TEXT. Could you please share that logic if you have so.

Thanks

Like 0 | Share



James Wood | Blog Post Author

September 18, 2013 at 8:54 pm

Ah, I missed the long text part. Please find an updated version of the code at the download link above. I made some wholesale changes to the code so that it's cleaner and more organized. Alas, the original demo was rather primitive. Let me know if you have any further questions. Thanks.

Like 0 | Share



Former Member

September 20, 2013 at 4:16 pm

James,

Thank you so much for the complete code.

Like 0 | Share



Former Member

September 19, 2013 at 1:55 am

Hi James,

This blog seems very helpful, so I'd like to start at the beginning of the series. Unfortunately the link in your first paragraph is broken. Any idea where I can find it?

Thanks,

Eric

Like 0 | Share



Former Member

September 19, 2013 at 8:21 am

It looks like the blogs were originally posted in personal space, and then moved to ABAP Development space.

Here are the correct links.

[Navigating the BOPF: Part 1 - Getting Started](#)[Navigating the BOPF: Part 2 - Business Object Overview](#)[Navigating the BOPF: Part 3 - Working with the BOPF API](#)[Navigating the BOPF: Part 4 - Advanced BOPF API Features](#)[Navigating the BOPF: Part 5 - Enhancement Techniques](#)[Navigating the BOPF: Part 6 - Testing & UI Integration](#)

Like 0 | Share



Former Member

October 9, 2013 at 3:17 pm

Hi James

Great blogs, starting to give me some understanding of BOPF.

I have a question though. I have a requirement for my client for a monster "super query" against the EHHSS\_INCIDENT business object which basically would allow them to query all Incidents by any combination of any selection criteria from any nodes!

I have already told them I think this is out of the question. However I want to at least be able to give them something. My main challenge is that it seems that Queries are attached to nodes, but what I really want is a cross-node query.

For example there is one node called "Persons involved" (table EHHSSD\_INC\_PINV) and another node called "Person Role" (table EHHSSD\_INC\_PROLE). This would allow me to query the persons involved in an Incident (selecting by John Smith for example) or the roles of

people involved in an incident (eg Witness). But what it does not allow me to do is to query the Incidents where John Smith is a Witness. To do that I have to use a foreign key relationship in a GUID on EHHSSD\_INC\_PROLE to point to the DB\_KEY on EHHSSD\_INC\_PINV.

So my main question is: Is it possible to do cross-node queries? If so how?

I thought about creating a database view as a join of the two tables, but then I don't know how to hook this database view onto the BO processing framework and how to attach queries to it. Or is the way of having a transient structure which is a join of the two tables & somehow hook this into database retrieval & queries.

Would really appreciate some guidance on this.

Thanks

Andy

Like 0 | Share



James Wood | Blog Post Author

October 10, 2013 at 3:21 pm

Hi Andrew,

To answer your question: yes. For your specific use case(s), I think the approach would be to create *custom queries*. So, for example, if you want to create a query in which you pull all incidents where John Smith is a witness, I think you'd simply want to create a custom query against the ROOT node. Here, you'd have a data structure to capture the selection criteria (e.g. person name and role) and a custom class to implement the query logic using a SQL JOIN. A decent example of this would be the SELECT\_BY\_OSHA\_CRIT query defined against the PERSON\_INVOLVED node. Hope this helps.

Thanks,

James

Like 0 | Share



Former Member

October 11, 2013 at 1:03 pm

Thanks James, good to know it is possible. I'll be giving it a try over the next few days.

Like 0 | Share



Zhenbo Wang

April 9, 2014 at 8:48 am

Great article, thanks. I am just beginning to like BOPF.

Like 0 | Share



Former Member

May 13, 2014 at 5:08 am

Hi James,

Thank you for publishing such an informative blog series to understand the BOPF.

I am facing one error in the program. Can you guide please?

I am trying to create a chemical using BO EHFND\_CHEMICAL using program. Referring the sample code listed in your blog.

I have passed ROOT node and BI\_CHEMICAL nodes. But I am getting error 'Mandatory node ROLE is missing'.

When I tried to add ROLE node , in the code its giving errors like 'Mandatory node ROLE is missing' OR can not create, source object does not exist'.

I went through the node structure for the BO. In it we have Root - > Revision -> Role.

So my query is how to pass information for ROLE node? We need to add REVISION node also?

If you can provide the code will be very helpful.

Like 0 | Share



**James Wood** | Blog Post Author

May 13, 2014 at 3:05 pm

Hi Shubhada,

Yes, you'll need to create REVISION node, too.

Thanks,

James

Like 0 | Share



Former Member

May 13, 2014 at 6:19 pm

Thanks James.

I have added REVISION node as below but still getting error 'Mandetary node ROLE missing'.

\*"Build the Revision node:

```
CREATE DATA lr_s_revision.
```

```
lr_s_revision->key    = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_revision->root_KEY = lr_s_root->key.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node      = IF_EHFND_CHM_C=>sc_node-revision.
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node = IF_EHFND_CHM_C=>sc_node-root.
```

```
<ls_mod>-association =
```

```
IF_EHFND_CHM_C=>sc_association-root-revision.
```

```
<ls_mod>-source_key = lr_s_root->key.
```

```
<ls_mod>-key      = lr_s_revision->key.
```

```
<ls_mod>-data      = lr_s_revision.
```

\* "Build the ROLE node:

```
CREATE DATA lr_s_role.
```

```
lr_s_role->key      = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_role->PARENT_KEY = lr_s_revision->key.
```

```
lr_s_role->ROOT_KEY = lr_s_root->key.
```

```
lr_s_role->chemical_role = '1'.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node      = if_ehfn_d_chm_c=>sc_node-role.
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node = if_ehfn_d_chm_c=>sc_node-revision.
```

```
<ls_mod>-association =
```

```
if_ehfn_d_chm_c=>sc_association-revision-role.
```



```
<ls_mod>-source_key = lr_s_revision->key.
```

```
<ls_mod>-root_key = lr_s_root->key.
```

```
<ls_mod>-key      = lr_s_role->key.
```

```
<ls_mod>-data      = lr_s_role.
```

Can you guide me please?

Like 0 | Share



**James Wood** | Blog Post Author

May 13, 2014 at 8:31 pm

Hi Shubhada,

What version of EHSM are you on? I'm looking at an EHSM 3.0 system with SP 4 installed and I don't see a REVISION or ROLE node available in EHFND\_CHEMICAL. These types of nodes are not uncommon to master data objects, so it wouldn't surprise me that they were added in a later release of the software. However, as I can't see the nodes myself, it's hard to speculate what the error condition might be. At a glance, your code above looks to be correct...

One thing I might suggest is to look closely at the contents of EO\_MESSAGE after you attempt to modify and/or save the BO. Here, I'd recommend scanning through the MT\_MESSAGE table to find the error message in question and see if the NODE\_KEY/VAL\_KEY fields are populated. This might give you more of a clue about where the error condition is emanating from. Hope this helps.

Thanks,

James

Like 0 | Share



Former Member

May 14, 2014 at 5:07 am

Hi James,

Thank you for the reply.

I am using SAP EHS Management Extension 4.0, release 400.

I will try to look at the MT\_MESSAGE as suggested by you.

Thank you.

Like 0 | Share



Former Member

August 14, 2014 at 4:47 am

Very good in detail information for techies.

Thank you very much and well done.

Regards,

Surender reddy

Like 0 | Share



**Bob Varghese**

August 25, 2014 at 1:14 pm

Hi James,

The above blog regarding BOPF is really good. ☐

Thanks for sharing your insight in BOPF.

Regards,

Bob.

Like 0 | Share



**Paul Hardy**

December 4, 2014 at 5:31 am

Mr.James,

Here is a very technical question about the mechanism whereby a number (like a customer number) gets turned into a GUID type key.

When I debug the SERVICE\_MANAGER->QUERY method I see that a fully dynamic SQL statement is being built up and then the database table queried using the customer number.

As there is no index on customer number I would expect a full table scan to occur, and the performance to be dreadful. Yet this does not seem to be the case - performance is OK and the ST05 trace did not say "full table scan" but some Oracle gobbledegook I had not seen before.

Is there some black magic at work here to get around the fact you are selecting on a field where there is no index?

Cheersy Cheers

Paul

Like 0 | Share



**James Wood** | Blog Post Author

December 4, 2014 at 2:03 pm

Hi Paul,

So am I correct in assuming that you're testing with the /BOBF/DEMO\_CUSTOMER business object demonstrated in this blog post? If so, I'm seeing that SAP has in fact created an index on the CUSTOMER\_ID field in the table behind the ROOT node (/BOBF/DM\_CST\_HDR). Are you seeing something different on your end?

In general, I would say that there's nothing real special going on with these node attribute queries. When you get past all of the dynamic code, the SQL queries executed by the BOPF runtime are basically OpenSQL as per usual.

Anyway, I hope this helps. If I'm off base with my analysis here, let me know a little more details about what you're testing with and I'll dig a little deeper.

Thanks,

James

Like 0 | Share



**Paul Hardy**

December 4, 2014 at 7:39 pm

As a test I had created my own object with a "number" field which is what the human in front of the computer would use to search for the object. An invoice number let us say.

As the primary key was the GUID and I deliberately did not put an index on the "number" then I expected the SQL trace to say "full table scan".

I actually got something like "ROW STOPKEY" which I think means the database looks at every single record in the table until it finds a match and then stops, which is in effect a full table scan.

I was just wondering if there was anything magic happening here, but it seems not.

This does throw into question the entire wisdom of having a database table with a GUID as the primary key - if you have an object where people are always going to search by number - invoices are a great example - then isn't having two indexes - the primary key and the index on the number - just a doubling up of resources?

I know in HANA world this is not going to matter, but realistically most people are not going to be there any time soon.

Cheersy Cheers

Paul

Like 0 | Share



**James Wood** | Blog Post Author

December 4, 2014 at 7:56 pm

Hi Paul,

OK, I'm with you now. You make a good point here on the wisdom of using GUIDs vs. semantic keys. In standalone environments, I frequently find this approach to be painful to work with (CRM comes to mind). In the dynamic world of the BOPF though, I think that the choice to use GUIDs actually makes a lot of sense. Being able to traverse from node to node by joining on `PARENT.DB_KEY = CHILD.PARENT_KEY` makes it very easy to build generic BO query frameworks where the performance is actually quite good. The primary overhead is when you hit the header table which would normally require an index on the semantic key. I suppose anytime you build a framework like this, there's going to be some overhead, but in my mind, what they have here is pretty manageable. Anyway, my two cents.

Thanks,

James

Like 0 | Share



Former Member

January 5, 2015 at 6:05 pm

Hi James

I am new to BOPF, and trying to set up a condition in TM using BOPF. Standard SAP provide a BO /SCMTMS/SUPPLIER which is a Master data Object. I can read a Carrier in run time using this. There is another BO /SCMTMS/TOR, where I can read the data from a freight order for example Customer (in ConsigneeID field).

So when in SAP TM, I select a carrier to be assigned to a Freight Order, I can read Carrier separately using /SCMTMS/SUPPLIER and Customer from the FO separately under /SCMTMS/TOR. Once the carrier is assigned and FO is saved, I can read the carrier under /SCMTMS/TOR as well under TSP\_ID but before that TSP\_ID is blank.

My requirement is to read the "carrier to be assigned" under /SCMTMS/TOR before FO is saved, so that I can check a condition between customer and carrier before saving it. In other words I want to read the Partner value of /SCMTMS/SUPPLIER (Master data Object) in /SCMTMS/TOR in Business Process Object. Is this feasible? How to achieve this. Looking forward for your response. Thanks for the Help.

Regards

Pankaj

Like 0 | Share



James Wood | Blog Post Author

January 6, 2015 at 12:37 am

Hi Pankaj,

I haven't worked with TM before, nor do I have access to a TM system, so I can only speculate on some of this. Some questions:

1. Am I correct in assuming that the /SCMTMS/SUPPLIER BO is linked to the /SCMTMS/TOR BO via a cross-business object (XBO) association?
2. Is TSP\_ID a (transient) attribute defined somewhere underneath the /SCMTMS/TOR BO node hierarchy?

If my assumptions above are correct, I expect that you should be able to back track from the XBO association class to figure out how the two BOs are physically linked. If the carrier's being identified before the save event occurs, I'd expect that you'd be able to get your hands on the foreign key to the carrier somewhere inside the /SCMTMS/TOR BO. From here, you may need to enhance/create a determination to preemptively fill the TSP\_ID field using a SQL query based on the selected carrier key.

Again, I'm sort of flying blind here, so let me know if I'm off base or need further clarification. Hope this helps.

Thanks,

James

Like 0 | Share





Former Member  
January 6, 2015 at 6:38 am

Hi James,

Thanks for your reply..

For the ease of understanding you can consider Freight order as a Sales Order and Carrier as a TSP (transport service provider partner which is not yet enter in the SO) and consignee as a ship to party.. Suppose you have saved the sales order with out the partner.. So you can see the Order number in VBAK table, you can see the Ship to party also in VBPA. But Since the carrier partner TSP is not yet assigned in the SO, it will be not be there in VBPA, although it exists as a master data in LFA1 table.

Now Consider LFA1 as /SCMTMS/SUPPLIER BO which is just a master data, and VBAK/PA as /SCMTMS/TOR BO which is the transaction data. When I pass the SO number in VBPA table, I can read ship to party, and when I pass the carrier number in LFA1, I can read the Carrier number from there.

Similarly I am using a data access determination using /SCMTMS/SUPPLIER (~LFA1) in a condition Cond1 and dad using /SCMTMS/TOR (~VBPA) in a condition Con2, when I pass the Carrier in cond1, I can read the carrier there and when I pass the freight order number in cond2, I can read ship to (consignee) in cond2, but since they are read in two different condition, I am not able to do some logical operations on them..

So I want to read both (Carrier to be assigned) and the Consignee under one condition. To do so I am trying to create a dad for /SCMTMS/SUPPLIER, and dad for /SCMTMS/TOR under same condition. Technically its not possible to so I am trying to read the /SCMTMS/SUPPLIER in /SCMTMS/TOR using some association and data crawler.

The TSP\_ID field stores the carrier for a freight order and is directly under the ROOT node of TOR BO. But it can be read only once the Carrier is entered and freight order is saved with it.

/SCMTMS/SUPPLIER is a master data BO, and stores the Carrier under ROOT node in Partner field.

I tried to find an association in Trxn BOPF for /SCMTMS/TOR and I could find an association named BO\_TSP\_ROOT, But I am not sure if it links with /SCMTMS/SUPPLIER or not, don't know how to check it.

I am looking for your help to find out more insight about association, and how to see what how two BO nodes associate.. and In case there in no association, is there any mechanism to read the Master data from a Master data node into the Business process Object in run time?

Sorry for such a lengthy post, I appreciate your help and patiently helping me out here.

Like 0 | Share



**James Wood** | Blog Post Author

January 6, 2015 at 3:03 pm

Hi Pankaj,

This all makes logical sense. One question though: in your description above you mention that the carrier is not yet assigned to the freight order. Assuming that's the case, I'm curious to understand *when* the condition you're building is supposed to fire? Am I correct in assuming that you want this to start kicking in at the point when the carrier's assigned but before the freight order's saved?

Anyway, can you send some screenshots in BOPF of the /SCMTMS/TOR BO? Looking specifically for screenshots with the expanded node hierarchy, association definitions, etc. That would help point you in the right direction I think.

Thanks,

James

Like 0 | Share



Former Member

January 7, 2015 at 9:02 am

*Display Business Object /SCMTMS/TOR, Active Version*

Business Object Detail Browser	Description
<ul style="list-style-type: none"> <li> <ul style="list-style-type: none"> <li>Node Structure</li> <li> <ul style="list-style-type: none"> <li>ROOT               <ul style="list-style-type: none"> <li>ATTACHMENTFOLDER</li> <li>CC_CHG_TR</li> <li>CHARGE_DISTRIBUTION</li> <li>CONFIRMATIONHISTORY</li> <li>CUSTOMS_ACTIVITY</li> <li>DIRECT_SHIPMENT_OPTIONS</li> <li>DOCREference</li> <li>EXECUTIONINFORMATION</li> <li>EXECUTIONINFORMATION_TR</li> <li>HANDLING_CODE</li> <li>HOUSE_SHIPMENT</li> <li>ITEM_TR</li> <li>LCADDRESS</li> <li>ORGUNIT_BP_RESTRICTION</li> <li>OVERVIEW</li> <li>PARTY</li> <li>RANKINGLIST</li> <li>STOP</li> <li>SUMMARY</li> <li>TENDERING</li> <li>TEXTCOLLECTION</li> <li>TRANSPORTCHARGES</li> <li>TRANSPORTCHARGES_INTERNAL</li> <li>BO_BUS_SHARE_REF</li> <li>CFIR_ITEM</li> <li>CFIR_ROOT</li> <li>SFIR_ROOT</li> </ul> </li> </ul> </li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Transportation Order</li> <li>Root Node</li> <li>Attachment Folder</li> <li>ChangeController: Change Information</li> <li>Subcontracting Confirmation History</li> <li>Customs Activity</li> <li>Alternatives for direct shipments</li> <li>Document Reference</li> <li>Execution Data</li> <li>Combined Execution Information</li> <li>Handling Codes</li> <li>House Shipment / House B/L node</li> <li>Items of the Transportation Order</li> <li>Letter of Credit Conform Address</li> <li>ORGUNIT and BP restrictions</li> <li>transient overview of logistical data</li> <li>Parties on Root node level</li> <li>Ranking List</li> <li>Stop</li> <li>Additional Info for TOR Root</li> <li>Tendering</li> <li>Text Collection</li> <li>Transportation Charges</li> <li>Internal Transportation Charges</li> <li>Reference Node in Business Share</li> <li>Reference incase of Consoles</li> <li>CFIR root node</li> <li>SFIR root node</li> </ul>

Business Object Detail Browser	Description
INSTRUCTION_INSTRUCTIONS	BO Instruction – Instructions node
OIAC_PROF_ROOT	Auto-Confirmation Profile for Org. Int.
PLAN_PROF_ROOT	BO Planning Profile
SFIR_ITEM	BO SFIR, node item
TRQ_ITEM	Item node of Transportation Request
TRQ_ROOT	TRQ root node
TSP_ROOT	Supplier Root Representation for the TSP
BO_WBN_CONSUMED	Reference to BO Waybill Consumed node
BO_WBN_ROOT	Reference to BO Waybill root node
BUPA_ROOT	Root node of Business Partner
CHANGE_DOCUMENT	Change History
OUTPUT_HISTORY	Output History
TAL_BUCKET	TAL Buckets loaded by this TOR
TAL_BUCKET_LOAD	TAL Bucket: Loading reference objects
Node Elements	
ATTACHMENTFOLDER	Attachment Folder
BO_BUS_SHARE_REF	Reference Node in Business Share
BO_CFIR_ITEM	Reference in case of Consoles
BO_FO_SCHEDULE_DEP_LOC	Departure Location in Schedule
BO_MATERIAL	Root Node of BO Material
BO_SFIR_ITEM	BO SFIR item
BO_TRQ_ITEMCONTENTID	Content IDs of referenced TRQ Item
BO_WBN_CONSUMED	Reference to BO Waybill Consumed node
BO_WBN_ROOT	Reference to BO Waybill root node
BUPA_BO_ITEMPTY	Item Business Partner
BUPA_ROOT	Root node of Business Partner
CC_CHG_TR	ChangeController: Change Information
CFIR_ITEM	Reference in case of Consoles
CFIR_ROOT	CFIR root node
CHANGE_DOCUMENT	Change History

Display Business Object /SCMTMS/TOR, Active Version

Business Object Detail Browser

Business Object Detail Browser	Description
▷ RANKINGLIST	Ranking List
▷ ROOT	Root Node
▷ Node Categories	
▷ Associations	
▷ ACTIVE_TENDERING	Navigate to active tendering
▷ ALL_HBL_NODES	All HBL nodes for FUs used in the TOR
▷ ALL_TENDERINGS	Association to all tenderings
▷ ASSIGNED_FUS	All directly or indirectly assigned FUs
▷ ASSIGNED_TVS	All assigned Transportation Units
▷ ATTACHMENTFOLDER	
▷ BO_BUS_SHARE	Business Share Link
▷ BO_CFIR_ITEM	Reference incase of Consoles
▷ BO_CFIR_ROOT	Returns all linked CFIR roots for TOR
▷ BO_FCI_ROOT	Association to FCI root
▷ BO_FDI_IN_DISP_ROOT	FDI with status In Dispute
▷ BO_FDI_ROOT	Freight Dispute Invoice
▷ BO_INSTRUCTION_INSTRUCTIO	Assoc. to BO Instruction - Instructions
▷ BO_OIAC_PROF_ROOT	Assoc. to Auto-Conf Profile
▷ BO_PLAN_PROF_ROOT	Assoc. to Planning Profile
▷ BO_SFIR_ITEM	Assoc. to SFIR item
▷ BO_SFIR_ROOT	Returns all linked SFIR roots for TOR
▷ BO_TRQ_ITEM	Assoc. to TRQ items
▷ BO_TRQ_ITEM_FU_ITEMS	Returns the items (linked by FU-Items)
▷ BO_TRQ_ROOT	Returns all linked TRQ roots for TOR FUs
▷ BO_TRQ_ROOT_ALL	Returns all linked TRQ roots for TOR
▷ BO_TSP_ROOT	Association to TSP Root
▷ Association Parameter Value 5	
▷ BO_WBN_CONSUMED	Association to WBN Consumed
▷ BO_WBN_ROOT	Association to WBN Root

Association

Association Name: BO\_TSP\_ROOT

Description: Association to TSP Root

Association Settings

Association Type: A Association

Association Category: X Cross Business Object association

Cardinality: 1 Cardinality 1..1

Source Node: 80E0ED0AOC021DDE8CE07D85DFAD0818 ROOT

Associated Node: 80E0ED0AOC021DEE858A2365EA911D8F TSP\_ROOT

Resolving Node: 1 Resolve by Source Node

Assoc. Change: 1 written by BOPF

Implementation

Association Class: /BOBF/CL\_LIB\_C\_CROSS\_BO

Filter Structure

Test Data

Test Data Container

Variant

Hi James

Actually its related to one Incompatibility setting, where when I select a Carrier and Freight Order, System checks the Carrier and Consignee (assigned in FO), and based on the condition result either allow or gives error. Standard SAP has given two separate condition for Carrier and FO, hence I have some limitation. I am trying to club both under one condition. My guess was that BO\_TSP\_ROOT can be one association, but somehow its not working as its not the Master data BO..

Thanks



James Wood | Blog Post Author

January 7, 2015 at 8:59 pm

Like 0 | Share

Can you also please send me a screenshot of the BO\_TSP\_ROOT association (highlighted above) when you double-click on it? In that definition, you should get a sense for how these two BOs are related. From here, perhaps we can backtrack and see if we can artificially build a linkage to satisfy your condition.

Thanks,

James

Like 0 | Share



Former Member

January 8, 2015 at 11:01 am

Hi James

The association is already there in the picture, If you click open it. Its bigger image hence not visible comment box..

Regards

Pankaj

Like 0 | Share



James Wood | Blog Post Author

January 8, 2015 at 2:51 pm

Yes, but what we need is the *details* around the association. For instance, is there an association class defined? On the Association Binding tab, what do the attribute bindings look like? To get where you want to go, you'll need to figure out how to hoist the carrier ID up to a point where you can access it in your condition. So, you may have to create another determination to achieve this which utilizes similar logic to the association definition.

Thanks,

James

Like 0 | Share



Former Member

January 9, 2015 at 6:29 am

Hi James

Thanks once again for helping me. I am adding some screenshots around association.
























The association class is /BOBF/CL\_LIB\_C\_CROSS\_BO

Please let me know If you are looking for anything else.

























































































Regards



























Pankaj





























































Association						
Association Binding						
Property Change Trigger						
Association Attribute Binding: BO_TSP_ROOT						
Association Binding Catego...	Source Node A...	Relational Operator	Binding Value From	Binding Value To	Constants Interface	Dat...
Cross-BO Binding Hint	TSP					

Association Property Change Trigger	Create	Update	Delete	Description
▼  ROOT		<input type="checkbox"/>		Root Node
 STOP~ASSIGNED_CAPA_TOR_ROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stop
 STOP_SUCCESOR~CAPAROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Successor of a Stop
 HOUSE_SHIPMENT~INCLUDED_REQ_TOR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	House Shipment / House B/L node
 ITEM_TR~REF_ORIG_ROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Items of the Transportation Order
 ITEM_TR~REF_TOR_ROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Items of the Transportation Order
 ITEM_TR~TOR_ITEM_TO_SRVO_ROOT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Items of the Transportation Order
 ORGUNIT_BP_RESTRICTION~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ORGUNIT and BP restrictions
 HANDLING_CODE~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Handling Codes
 EXECUTIONINFORMATION_TR~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Combined Execution Information
 TEXTCOLLECTION~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Text Collection
 CC_CHG_TR~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ChangeController: Change Information
 TRANSPORTCHARGES~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Transportation Charges
 ATTACHMENTFOLDER~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Attachment Folder
 PARTY~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Parties on Root node level
 RANKINGLIST~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Ranking List
 ITEM_TR~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Items of the Transportation Order
 STOP~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Stop
 CONFIRMATIONHISTORY~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Subcontracting Confirmation History
 SUMMARY~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Additional Info for TOR Root
 EXECUTIONINFORMATION~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Execution Data
 DOCREFERENCE~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Document Reference
 TENDERING~TO_PARENT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Tendering



  PARTY~TO_ROOT				Parties on Root node level
  STOP~TO_ROOT				Stop
  STOPTEXTCOLLECTION				StopTextCollection
  EXECUTIONNOTES				Execution Notes
  TENDERINGSTOPDATE				Tendering Stop Date
  TENDERINGRESPONSE				Tendering Response
  TENDERINGREQUEST				Tendering Request
  TENDERINGSTEP				Tendering Step
  STOP_SUCCESOR~TO_ROOT				Successor of a Stop
  EXECUTIONINFORMATION~TO_ROOT				Execution Data
  CONFIRMATIONHISTORY~TO_ROOT				Subcontracting Confirmation History
  DOCREference~TO_ROOT				Document Reference
  SUMMARY~TO_ROOT				Additional Info for TOR Root
  TENDERING~TO_ROOT				Tendering
  CUSTOMS_GROUP				Customs Group
  CUSTOMS_ACTIVITY~TO_ROOT				Customs Activity
  DIRECT_SHIPMENT_OPTIONS~TO_ROOT				Alternatives for direct shipments
  HOUSE_SHIPMENT_ITEM				House shipment items
  HOUSE_SHIPMENT~TO_ROOT				House Shipment / House B/L node
  TRANSITCOUNTRY				Transit countries of a stage
  CUSTOMS_GROUP_ITEM				Customs Group Item
  OVERVIEW~TO_ROOT				transient overview of logistical data

  CUSTOMS_ACTIVITY~TO_PARENT				Customs Activity
  DIRECT_SHIPMENT_OPTIONS~TO_PARENT				Alternatives for direct shipments
  HOUSE_SHIPMENT~TO_PARENT				House Shipment / House B/L node
  OVERVIEW~TO_PARENT				transient overview of logistical data
  TRANSPORTCHARGES_INTERNAL~TO_PARENT				Internal Transportation Charges
  CHARGE_DISTRIBUTION~TO_PARENT				
  LCADDRESS~TO_PARENT				Letter of Credit Conform Address
  ITEMPARTYADDRESS				Item Party Address
  ORGUNIT_BP_RESTRICTION~TO_ROOT				ORGUNIT and BP restrictions
  ITEMDOCREference				Item Document Reference
  HANDLING_CODE~TO_ROOT				Handling Codes
  PARTYLCADDRESS				Party Letter of Credit Conform Address
  EXECUTIONINFORMATION_TR~TO_ROOT				Combined Execution Information
  TEXTCOLLECTION~TO_ROOT				Text Collection
  ATTACHMENTFOLDER~TO_ROOT				Attachment Folder
  TRANSPORTCHARGES~TO_ROOT				Transportation Charges
  TENDERINGRESPONSEATTACHMENTFOL				Tendering Response Attachment Folder
  TENDERINGREQUESTATTACHMENTFOLD				Tendering Request Attachment Folder
  RANKINGLIST~TO_ROOT				Ranking List
  ITEM_TR~TO_ROOT				Items of the Transportation Order
  ITEMTRDANGGOODSINFO				tr. Item Dangerous Goods Information
  CC_CHG_TR~TO_ROOT				ChangeController: Change Information
  TENDERINGRESPONSENOTES				Tendering Response Notes

  CUSTOMS_GROUP_ITEM				Customs Group Item
  OVERVIEW~TO_ROOT				transient overview of logistical data
  TRANSPORTCHARGES_INTERNAL~TO_ROOT				Internal Transportation Charges
  CHARGE_DISTRIBUTION~TO_ROOT				
  ITEMPARTY				Item Party
  ITEMTEXTCOLLECTION				ItemText Collection
  SEAL				Seals
  LCADDRESS~TO_ROOT				Letter of Credit Conform Address
  EXECUTIONATF				Execution Attachments
  ITEMCCODE				Commodity Code
  STOP_SUCC_DOCPREFERENCE				Stage Document Reference
  ITEMCONTENTID				Item Content Identification

Association

Association Binding

Property Change Trigger

Association Type	A Association	
Association Category	X Cross Business Object association	
Cardinality	1 Cardinality 1:O..1	
Source Node	80EOEDOAO021DDE8CE07DB5DFAD0818 ROOT	
Associated Node	80EOEDOAO021DEEB58A2365EA911D8F TSP_ROO...	
Resolving Node	1 Resolve by Source Node	
Assoc. Change	1 written by BOPF	

Implementation

Association Class


Filter Structure

Test Data

Test Data Container

Variant

Administrative Data



James Wood

SAP Blog Post Author

10.11.2013 9 15:47:01

Created

January 13, 2015 at 5:04 pm

10.12.2013 20:36:11

Hi Pankaj  
Like 0 | Share

Given the way this association is defined, I'm thinking that you may have to get clever with this. I'm thinking something along the lines of the following:

1. Implement some enhancement logic to intercept the carrier assignment event and store the selected carrier ID in a shared memory object (which internally uses a hash table to associate the carrier ID with the corresponding FO).
2. Create a transient attribute on the root node of the freight order BO to expose the carrier ID.
3. Create a custom determination to populate the transient attribute with the carrier ID value from shared memory.

I think this should allow you to access the carrier ID from your condition record before the FO is saved. What do you think?

Thanks,

James

Like 0 | Share



Former Member

April 5, 2018 at 7:52 pm

Hello Pankaj,

I am a SAP Technical guy new to TMS...We have a bunch of inbound EDI coming into TMS...can you tell me if TMS has IDOC's built in it like ECC to handle these Inbound EDI or else how are they posted in TMS?

Any help would be appreciated

Thanks

Ram

Like 0 | Share



Former Member

January 14, 2015 at 10:23 am

Hi James

I am not very strong in technical area specially in BOPF as I mostly work in functional side only, but If I translate what I understood in my language, its a 3 step process

1. - I need to read the Carrier data in a custom Table (at the time of carrier creation event, the carrier will be stored in /scmtms/supplier as well as in this custom table)

2. - Maintain a custom defined field on BO node of Freight order for carrier.

and

3. - then read from there at the run time while saving the FO using the custom determination..

Please let me know if this is correct understanding.. So could you please help with the steps to do it. I will try and see if this works. Also I can use this idea in other problems as well.

I also have one basic query.. Once I find the association for a BO node, how can I find out what other BO nodes its associated to..

Thanks

Pankaj

Like 0 | Share



**James Wood** | Blog Post Author

January 14, 2015 at 2:43 pm

Hi Pankaj,

My thought with point #1 was to capture the event when the carrier is associated with the freight order (but before the FO is saved). Based on what you commented earlier (see below), I gathered that this was the gap you were struggling with:

figuring out how to read the carrier ID in a condition before the FO is saved. The logic described above was intended to provide you with a separate field which makes it easy to link up the FO and carrier data from within the FO business object. Am I off base here?

*...Once the carrier is assigned and FO is saved, I can read the carrier under /SCMTMS/TOR as well under TSP\_ID but before that TSP\_ID is blank...*

Thanks,

James

Like 0 | Share



Former Member

January 15, 2015 at 11:29 am

No James, you are absolutely right.. I think I misunderstood pt. 1.. So how to proceed on this?

Thanks and regards

Pankaj

Like 0 | Share



James Wood | Blog Post Author

January 15, 2015 at 2:58 pm

Hi Pankaj,

The next step is to start writing code. Perhaps there's a developer on your project you can work with to take this concept to realization. Best of luck with the project.

Thanks,

James

Like 0 | Share



Former Member

January 16, 2015 at 8:37 am

Sure James.. I will check with the development team here..

Thanks a lot for your valuable suggestions on this topic.. I will keep you updated if this works.

Thanks and Regards

Pankaj

Like 0 | Share



James Wood | Blog Post Author

January 12, 2015 at 3:38 pm

Hi Shakeel,

The issue with your code is in the way you're creating the PERSON\_INVOLVED record. For both records, you're mapping SY-UNAME to the PERSON\_INVOLVED.ID field. To create separate records, you need to map different IDs for each distinct person. Here, you have three different types

to choose from:

Employee Types (A + {pernr})

Business Partner Types (B + {bupa})

User Types (D + {user ID})

Without knowing a ton about your use case, it would seem that if all you have to go on is contact information, you probably will have to either use that information to look up one of the three person types listed above or create an ad hoc business partner on the fly. Regardless of the path you take, the resultant ID is what you would plug into the ID field. Do that, and I think you're on the right track.

Thanks,

James

Like 0 | Share



**Md Shakeel Ahmed**

January 13, 2015 at 8:11 am

Hi James,

Thanks for quick replay.

I have tried by your above inputs and know two Involved person are getting created.



But how to create the REPORTING PERSON under the same INVOLVED\_PERSON node. I.e mean who is creating the incident.

I am trying to create the three INVOLVED PERSONS

1.Reporting Person

2.Injured Person

3. Witness Person

Thanks a lot.

Regards,

Shakeel.

Like 0 | Share



**James Wood** | Blog Post Author

January 13, 2015 at 4:55 pm

The reporting person would be created just like all the others. In this case though, you'd probably want to use the CL\_EHFND\_PARTY\_PROXY class' CONVERT\_USER\_NAME\_TO\_PARTY\_KEY() method to convert SY-UNAME into an EHSM party key. Then, plug that key into the ID field and assign the reporting person role in the PERSON\_ROLE node. Thanks.

Like 0 | Share



**Md Shakeel Ahmed**

January 19, 2015 at 9:35 am

Hi James,

Thanks for reply.

I have followed the above steps and resolved my issue.

I have another question to ask about the delegated node like 'NEAR\_MISS\_DESC' and 'INJURY\_ILLNESS\_BP\_DESC' in EHHSS\_INCIDENT.

I wrote this code to create the injury description but it is not updating.

```
CREATE DATA lr_s_inj_info.
```

```
lr_s_inj_info->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_inj_info->oc_inc_type = 'EHHSS_OIT_ACC_ON_WAY'.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node      = if_ehhss_inc_c=>sc_node-person_inj_info.
```

```
<ls_mod>-change_mode  = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node  = if_ehhss_inc_c=>sc_node-person_involved.
```

```
<ls_mod>-association  = if_ehhss_inc_c=>sc_association-person_involved-person_inj_info.
```

```
<ls_mod>-root_key    = lr_s_root->key.
```

```
<ls_mod>-source_key   = lr_s_per_inv->key.
```

```
<ls_mod>-key          = lr_s_inj_info->key.
```

```
<ls_mod>-data         = lr_s_inj_info.
```

```
CREATE DATA lr_s_injury_illness.
```

```
lr_s_injury_illness->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_injury_illness->inj_ill = 'EHHSS_ILLC_INJ'.
```

```
*  lr_s_INJURY_ILLNESS->TYPE = 'EHHSS_OIT_ACC_ON_WAY'.
```

```
lr_s_injury_illness->type_desc = desc. " Description
```

```
lr_s_injury_illness->type_desc = 'DESCR'.
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node        = if_ehhss_inc_c=>sc_node-injury_illness.
```

```
<ls_mod>-change_mode  = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node  = if_ehhss_inc_c=>sc_node-person_inj_info.
```

```
<ls_mod>-association  = if_ehhss_inc_c=>sc_association-person_inj_info-injury_illness.
```

```
<ls_mod>-root_key     = lr_s_root->key.
```

```
<ls_mod>-source_key   = lr_s_inj_info->key.
```

```
<ls_mod>-key      = lr_s_injury_illness->key.
```

```
<ls_mod>-data      = lr_s_injury_illness.
```

```
CREATE DATA lr_s_root_txt.
```

```
lr_s_root_txt->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
APPEND INITIAL LINE TO lt_mod ASSIGNING <ls_mod>.
```

```
<ls_mod>-node      = if_ehhss_inc_c=>sc_node-INJURY_ILLNESS_DESC.
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node = if_ehhss_inc_c=>sc_node-INJURY_ILLNESS.
```

```
<ls_mod>-association = if_ehhss_inc_c=>sc_association-INJURY_ILLNESS-injury_illness_desc.
```

```
<ls_mod>-root_key   = lr_s_root->key.
```

```
<ls_mod>-source_key = lr_s_INJURY_ILLNESS->key.
```

```
<ls_mod>-key        = lr_s_root_txt->key.
```

```
<ls_mod>-data        = lr_s_root_txt.
```

```
create data lr_s_INJURY_ILLNESS_DESC.
```

```
lr_s_INJURY_ILLNESS_DESC->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_INJURY_ILLNESS_DESC->TEXT_EXISTS_IND = 'X'.
```

APPEND INITIAL LINE TO It\_mod ASSIGNING <ls\_mod>.

<ls\_mod>-node = if\_ehhss\_inc\_c=>sc\_node-INJURY\_ILLNESS\_DESC.

<ls\_mod>-change\_mode = /bobf/if\_frw\_c=>sc\_modify\_create.

<ls\_mod>-source\_node = if\_ehhss\_inc\_c=>sc\_node-INJURY\_ILLNESS.

<ls\_mod>-association = if\_ehhss\_inc\_c=>sc\_association-INJURY\_ILLNESS-INJURY\_ILLNESS\_DESC.

<ls\_mod>-root\_key = lr\_s\_root->key.

<ls\_mod>-source\_key = lr\_s\_INJ\_INFO->key.

<ls\_mod>-key = lr\_s\_INJURY\_ILLNESS\_DESC->key.

<ls\_mod>-data = lr\_s\_INJURY\_ILLNESS\_DESC.

"Create the TEXT node:

CREATE DATA lr\_s\_text.

lr\_s\_text->key = /bobf/cl\_frw\_factory=>get\_new\_key( ).

lr\_s\_text->TEXT\_TYPE = 'DESCR'.

APPEND INITIAL LINE TO It\_mod ASSIGNING <ls\_mod>.

<ls\_mod>-node = lo\_driver->mo\_bo\_conf->query\_node( iv\_proxy\_node\_name = 'INJURY\_ILLNESS\_DESC.TEXT' ).

<ls\_mod>-change\_mode = /bobf/if\_frw\_c=>sc\_modify\_create.

```
<ls_mod>-source_node = if_ehhss_inc_c=>sc_node-injury_illness_desc.
```

```
<ls_mod>-source_key = lr_s_INJURY_ILLNESS_DESC->key.
```

```
<ls_mod>-root_key = lr_s_root->key.
```

```
<ls_mod>-key = lr_s_text->key.
```

```
<ls_mod>-data = lr_s_text.
```

```
<ls_mod>-association =
```

```
lo_driver->mo_bo_conf->query_assoc(
```

```
iv_node_key = if_ehhss_inc_c=>sc_node-injury_illness_desc
```

```
iv_assoc_name = 'TEXT' ).
```

"Create the TEXT\_CONTENT node:

```
CREATE DATA lr_s_txt_cont.
```

```
lr_s_txt_cont->key = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_txt_cont->TEXT = 'Text for Injury / Illness Description'.
```

APPEND INITIAL LINE TO lt\_mod ASSIGNING <ls\_mod>.

```
<ls_mod>-node = lo_driver->mo_bo_conf->query_node( iv_proxy_node_name =  
'INJURY_ILLNESS_DESC.TEXT_CONTENT' ).
```

```
<ls_mod>-change_mode = /bobf/if_frw_c=>sc_modify_create.
```

```
<ls_mod>-source_node = if_ehhss_inc_c=>sc_node-injury_illness_desc.
```

```
<ls_mod>-source_key = lr_s_INJ_INFO->key.  "lr_s_text->key.
```

```
<ls_mod>-root_key  = lr_s_root->key.
```

```
<ls_mod>-key      = lr_s_txt_cont->key.
```

```
<ls_mod>-data     = lr_s_txt_cont.
```

```
<ls_mod>-association =
```

```
  lo_driver->mo_bo_conf->query_assoc(
```

```
    iv_node_key = if_ehhss_inc_c=>sc_node-injury_illness_desc
```

```
    iv_assoc_name = 'TEXT_CONTENT' ).
```

Thanks,

Shakeel

Like 0 | Share



**James Wood** | Blog Post Author

January 19, 2015 at 3:40 pm

For near misses, you can fill in the description field directly in the NEAR\_MISS node using the DESC\_TEXT field. Behind the scenes, BOPF determinations will copy the text into the subordinate node automatically.

For injury illness, the field is BP\_DESC\_TEXT in the INJURY\_ILLNESS node.

Hope this helps.

Thanks,

James

Like 0 | Share



**Md Shakeel Ahmed**

January 20, 2015 at 3:45 pm

Hi James,

Thank You.

Now texts are getting created.

Thanks & Regards,

Shakeel

Like 0 | Share



**Md Shakeel Ahmed**

January 21, 2015 at 2:21 pm

Hi James,

I am trying to attach the documents like images, videos, .doc and .xls file to sap but it is uploading.

Here is my code.

```
*****Attach file to Node ATT_DOCUMENT
```

```
data: lv_filesize TYPE sdok_fsize.
```

```
CREATE DATA lr_s_att_document.
```

```
describe field content length lv_filesize in byte mode.
```

```
lr_s_att_document->key    = /bobf/cl_frw_factory=>get_new_key( ).
```

```
lr_s_att_document->key_ref = lr_s_root->key.
```

```
lr_s_att_document->FILE_SIZE = lv_filesize .
```

```
lr_s_att_document->FORM_NAME = 'INC_INFO_WITNESS'.
```

```
lr_s_att_document->MIME_CODE = '/SAP/PUBLIC/BOBF'.
```

```
lr_s_att_document->FILE_NAME = 'EHS_Image_file'.
```

```
lr_s_att_document->CONTENT  = 'C:\Users\Desktop\mobo_logo.png'.
```

APPEND INITIAL LINE TO lt\_mod ASSIGNING <ls\_mod>.

<ls\_mod>-node = if\_ehhss\_inc\_c=>sc\_node-ATT\_DOCUMENT.

<ls\_mod>-change\_mode = /bobf/if\_frw\_c=>sc\_modify\_create.

<ls\_mod>-source\_node = if\_ehhss\_inc\_c=>sc\_node-root.

<ls\_mod>-association = if\_ehhss\_inc\_c=>sc\_association-root-ATT\_DOCUMENT.

<ls\_mod>-source\_key = lr\_s\_root->key.

<ls\_mod>-key = lr\_s\_att\_document->key.

<ls\_mod>-data = lr\_s\_att\_document.

data: lt\_att\_doc\_key TYPE /bobf/t\_frw\_key,

IS\_ATTACHMENT type /BOBF/S\_ATF\_A\_CREATE\_FILE.

"Create the attachment record:

CALL METHOD lo\_driver->mo\_svc\_mgr->DO\_ACTION(

exporting

iv\_act\_key = if\_ehhss\_inc\_c=>sc\_action-att\_document-upload\_document

it\_key = lt\_att\_doc\_key ).

\* is\_parameters = is\_attachment ).

Please help me.

Thanks & Regards,

Shakeel Ahmed.

Like 0 | Share



James Wood | Blog Post Author

January 21, 2015 at 2:31 pm

Hi Shakeel,

Send me an e-mail and I can send you some sample code to look at.

Thanks,

James

<email address removed by moderator, it is already shared in SCN profile>

Like 0 | Share

Former Member



November 26, 2015 at 6:18 am

Hi James, Can you please share the sample code for Attachment folder? Thanks, Kp

Like 0

| Share

**Sanjeev Gowda**

May 6, 2020 at 12:34 am

Hi James,

Thanks for the wonderful blog and explanation . I am having issue with attaching document. Can you please share the sample code Thanks ...

Like 0 | Share

**Sanjeev Gowda**

May 5, 2020 at 10:40 pm

Hi, Were you able to resolve the issue of attaching the document . If so , can u please share the steps or code snippet to resolve the issue . thanks for your help

Like 0 | Share

**Former Member**

January 8, 2016 at 10:37 am

James, This blog very helpful for me to understand BOPF. Thank you.

Like 0 | Share

**Former Member**



October 21, 2016 at 2:13 pm

Thank you very much for this blog, god knows how long would I have to crawl through tons of generic code hadn't I found this here!

A commendable effort!

Like 0 | Share



**Vishwanath Gupta**

January 12, 2017 at 3:19 am

James,

Thanks a ton for detailed blogs on BOPF.

I was searching for some information around how buffer works in BOPF frame work like control of load of buffer, update to the buffer and to the DB from buffer etc..

Can you suggest any source of information on this topic.

Thanks in advance,  
Vishwa.

Like 0 | Share



**sudarshan rajam david**

May 4, 2018 at 1:40 pm

Hi [James Wood](#),

Great Blog, Thank you so much for it.

After reading the blog part 3 at the end, one question raised in me => I am eager to know what the point creating customer master via BOPF API, we can also use BAPI's for creating.

And If its just an example, then does we have to create screens and call the lcl\_demo, for creating Customer ID.

Please let me know, I am curious about it.

Thanks in Advance,

Sudarshan D

Like 0 | Share



**David Lawn**

June 14, 2018 at 11:39 am

Problem

Hi James Wood and thank you.

I have a problem trying to write to the DOCUMENT standard node of /BOBF/ATTACHMENT\_FOLDER delegated object under /SRMSMC/MO\_BUPA~ROOT\_ATTACHMENT\_FOLDER ("RAF").

You write: “However, if we had wanted to do so, we would have needed to create a separate service manager instance for the `/BOBF/DEMO_TEXT_COLLECTION` business object since the data within that node is defined by that delegated BO as opposed to the `/BOBF/DEMO_CUSTOMER` BO.”

So I understand I need a new instance of service manager to be able to work with `/BOBF/ATTACHMENT_FOLDER`.

I find

- I can read existing DOCUMENT entries using the same service manager instance I use for everything else (query `MO_BUPA`, `retrieve_by_association` to get to RAF, modify, ...) but I cannot write any new DOCUMENTs to my newly-created `MO_BUPA~RAF` node. No error messages but no documents. (I do do `get_content_key_mapping` to get association and node.)
- when I try to create a second service manager using

```
me->mo_svc_mgr_do =  
  /bobf/cl_tra_serv_mgr_factory=>get_service_manager(  
    /bobf/if_attachment_folder_c=>sc_bo_key ).
```

I get an error coming from `/BOBF/CL_TRA_SERV_MGR_FACTORY` saying

```
IF ls_confro_object-objcat = /bobf/if_conf_c=>sc_objcat_do.  
" It is not allowed to access a delegated object directly via a service manager  
" Only the host object may access its dependend objects.  
set_application_error( ).
```

So can you suggest the way? I seek to create new DOCUMENTs under my newly created RAF.

With best wishes

Like 0 | Share



Jacky D

August 18, 2021 at 12:08 pm

Hi Devid.

i ran into the same question, did you got any solution?

Like 0 | Share



**jugal singh**

September 21, 2018 at 3:16 pm

Hi James,

query option is not working for me.I don't see sc\_query attribute

not able to pass /bobf/if\_demo\_customer\_c=>sc\_query-root-select\_by\_attributes. and it's dumping here.

Please suggest how to achieve this.

Thanks in Advance...

Jugal

Like 0 | Share



**Jerry Zhang**

February 9, 2019 at 4:21 pm

Why in this demo do you use local class? If this class is create private, how can I try all the methods in it? Thank you!

Like 0 | Share



**Shivakrishna Eshakoyla**

July 24, 2019 at 6:07 am



Hi Zhang,

Declare CREATE\_CUSTOMER as public static method and access the method using class name. Since CREATE\_CUSTOMER method has self object created (lo\_driver), this will trigger constructor (private) and rest of the objects are instantiated (Transaction manager, Service manager and configuration manager).

regards,

Shiva Krishna E

Like 0 | Share



Ismail ElSayed

May 5, 2019 at 12:02 pm

Can you upload the program again ? because the exist link is not working

Like 0 | Share




Ka Ro

July 22, 2019 at 5:32 am

I have test to create new instance in bopf test environment (BOBT) for BO:/BOBF/DEMO\_CUSTOMER, and saved it with error displayed for input data. and after this , when i run bobt by BO:/SCMTMS/TOR [by alternative key :tor\_id] and input F0 to search data, then there is a short dump happened.

is there anyone know how to solve the error , thank u .

## Runtime Error – Description of Exception

 Long Text    Debugger

Category	ABAP programming error
Runtime Errors	MESSAGE_TYPE_X
ABAP Program	/BOBF/CX_FRW_FATAL=====CP
Application Component	BC-ESI-BOF
Date and Time	07/18/2019 10:16:16

### Short Text

The current application has triggered a termination with a short dump.

### What happened?

The current application program has detected a situation that should not occur. A termination with short dump has therefore been triggered by the key word MESSAGE (type X).

### Error analysis

Short text of the error message:  
 Internal application error of BO /SCMTMS/BUPA of component TM-MD-BP; see application info

Long text of the error message:  
 Technical information about the message:  
 Message class..... /BOBF/FRW  
 Number..... 013  
 Variable 1..... /SCMTMS/BUPA  
 Variable 2..... TM-MD-BP  
 Variable 3.....  
 Variable 4..... " "

## Trigger Location of Runtime Error

Program	/BOBF/CX_FRW_FATAL=====CP
Include	/BOBF/CX_FRW_FATAL=====CM005
Row	27
Module Type	(METHOD)
Module Name	MESSAGE_TYPE_X

## Source Code Extract (Source code changed)

Line	SourceCde
1	METHOD message_type_x.
2	"Check if we are within an AUnit test execution. In that case, gracefully
3	"abort the test class, but do not dump. Thus, it is easier to locate the
4	"test triggering the failure.
5	DATA lt_callstack TYPE abap_callstack.
6	DATA ls_callstack_entry TYPE abap_callstack_line ##needed.
7	DATA lv_error_detail TYPE string.
8	
9	CALL FUNCTION 'SYSTEM_CALLSTACK' IMPORTING callstack = lt_callstack.
10	LOOP AT lt_callstack INTO ls_callstack_entry WHERE mainprogram CP 'CL_AUNIT_TEST_CLASS*'
11	IF application_information IS NOT INITIAL.
12	lv_error_detail = application_information.
13	ELSEIF previous IS BOUND.
14	DATA(lo_previous) = previous.
15	WHILE lo_previous->previous IS BOUND.
16	lo_previous = lo_previous->previous.
17	ENDWHILE.
18	lv_error_detail = lo_previous->get_text( ).
19	ENDIF.
20	cl_abap_unit_assert=>fail( level = if_aunit_constants=>fatal

Like 0 | Share

**IRINEU AVANCO**

July 29, 2019 at 1:03 pm

Hello James ..

I am trying to download the source code of this article, as you mentioned , bu the link is not working.

Could you please kindly provide us with an alternative link to get access to this source code ?

Thank you !

Like 3 | Share

**Sudhanshu Sharma**

June 4, 2020 at 11:17 am

Hi,

Its been long time this article was posted. If possible please provide new link(s) to download to source code.

Thanks

Like 1 | Share

**Izabela Kotus**

September 29, 2020 at 7:50 am

Hi,

in addition, could you provide guidelines/examples of unit test classes for enabling access/modifications of BOPF objects?

Thanks

Like 0 | Share

**Find us on**

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences
Newsletter	Support