PUBLIC

How To Integrate the BCV Side Panel into a Web Dynpro Application

Applicable Releases:

SAP Business Context Viewer (BCV) 7.01 (Enhancement Package 5 for SAP ERP 6.0) or higher

Version 1.11

September 2024

Document History

Document Version	Description
1.0	First official release of this guide
1.01	Minor updates
1.10	Minor updates
1.11	Minor updates

Table of contents

1. Introduction 4
1.1. Preconditions 4
1.2. Effort Estimation 5
2. Plain WebDynpro ABAP application 5
2.1. Constraints and Preconditions 6
2.2. Side Panel Initialization 6
2.2.1. Web Dynpro Action 8
2.2.2. Web Dynpro UI Element Page Header 8
2.3. Side Panel Update (Tagging) 9
2.3.1. Tagging Service 10
3. Floorplan Manager Applications 12
3.1. WebDynpro Application Configuration 12
3.2. FPM Application Based on PLM UI Framework 13
3.2.1. Side Panel Initialization 13
3.2.2. Side Panel Update (Tagging) 14
3.3. FPM Overview Page Application using Generic User Interface Building Blocks 15
3.3.1. Side Panel Initialization 15
3.3.2. Side Panel Update (Tagging) 16
3.4. PM Application using Freestyle User Interface Building Blocks 16
3.4.1. Side Panel Initialization 16
3.4.1. Side Panel Update (Tagging) 18
4. Data Transfer 18
4.1. Tagging Service 19
4.1.1. Tag vs. Meaning 19
4.1.2. Tagging using Configuration 19
4.1.3. Example Data Transfer 20
4.2. Context Key 21
4.2.1. Creation of BCV Context Key 22
4.2.1. Namespace of Context Key vs. Configuration ID 24
4.1. System Alias 24
5. Additional Information 26
5.1. Further Reading 26
5.1.1. Information on SAP MDG on SAP S/4HANA 26
5.1.2. SAP Roadmap Explorer 26
5.1.3. Related Information 26
5.2. SAP Notes 26

1. Introduction

SAP Master Data Governance (MDG) provides business processes to find, create, change, and mark master data for deletion. It supports the governance of master data in a central hub and the distribution to connected operational and business intelligence systems.
The processes are workflow-driven and can include several approval and revision phases, and the collaboration of all users participating in the master data maintenance.
MDG offers change request (CR)-based processing of master data with integrated workflow, staging, approval, activation, and distribution.

This document describes how to integrate the BCV side panel with your Web Dynpro application (also known as a "hosting application", as it hosts the side panel).

There are different ways to integrate the BCV side panel, depending on whether the application uses Floor Plan Manager (FPM) or not.

You can integrate the BCV side panel in the following applications:

- "Plain" Web Dynpro ABAP Application
- FPM Application based on the PLM UI Framework
- FPM Overview Page (OVP) Application using Generic User Interface Building Blocks (GUIBB)
- FPM Application using freestyle User Interface Building Blocks (UIBB)

You follow the same procedure for all applications:

- Configure the usage of the BCV side panel
- Initialize the side panel
- Update the side panel with parameter values from your application (tagging)

1.1. Preconditions

You should consider the following preconditions before you start the side panel integration:

- You have installed the Business Suite Foundation Layer (SAP_BS_FND). Every ABAP Web Dynpro-based Business Suite application (from EHP5 on) can use BCV. Add the interface **`BS_BUSINESS_CONTEXT_VIEWER`** (from the package **`BS_REUSE`**) to the use access declaration of your structure package and the included sub-packages where necessary.
- The BCV business function **`/BCV/MAIN`** is active. You can check this using transaction SWF5. If the business function is not active, contact your system administrator using an IT/IBC message.
- You need basic BCV configuration settings, such as a context key and meanings, to inform the side panel about the hosting application and the parameters for which you want to see additional information. See section *Context Key*.

- You need the authorization to execute the BCV side panel and to configure BCV to your needs. For example, you can use the role **`SAP_BCV_ADMIN`** as a template for your user.

1.2. Effort Estimation

It should take you less than 30 minutes to implement the integration if you are familiar with the following:

- Web Dynpro ABAP and Floor Plan Manager
- The screens you want to integrate with BCV
- The key fields of the business object (for example, Material) the BCV queries will be based on

This implementation displays the ready-to-use side panel link with functionality in the upper right-hand corner.

If you are familiar with the concept of BCV (for example, search connectors, BCV queries, query views, overview, dashboard), you need an additional hour or less to do the configuration.

The necessary configuration is almost independent from the integration described in the following, for example, it can be done later or by the customer. However, without the configuration, no data is received or presented.

To be able to do this configuration, you need a data provider, for example, an InfoCube. Data providers are usually defined using the scenarios that should be implemented. We expect the definition of these scenarios and the determination of the appropriate data providers to take the most effort. This effort depends on your application only and is independent of BCV.

Part of the preparation to use BCV can include the provision of the data providers: definition of the InfoCube or InfoSet, definition of the template in Enterprise Search, or implementation of the BAPI adapter.

If a BAPI adapter is needed for data retrieval using BAPIs, RFC, or local call to the application, you need about one day to implement each adapter. This is working on the assumption that the necessary code is available on the application side.

2. Plain WebDynpro ABAP application

If your Web Dynpro application does not use FPM, the following is required for your application coding:

- Side panel initialization using the NetWeaver application programming interface (API) and action to open the side panel
- Side panel update (tagging)

2.1. Constraints and Preconditions

The UI of your application must be implemented in Web Dynpro ABAP.

To integrate the BCV side panel to your application, you need a view. The view must have a page header named PAGE_HEADER.

If your application does not contain its own view, for example, because you are reusing the personal object worklist (POWL) standard view, you can use a workaround such as the following:

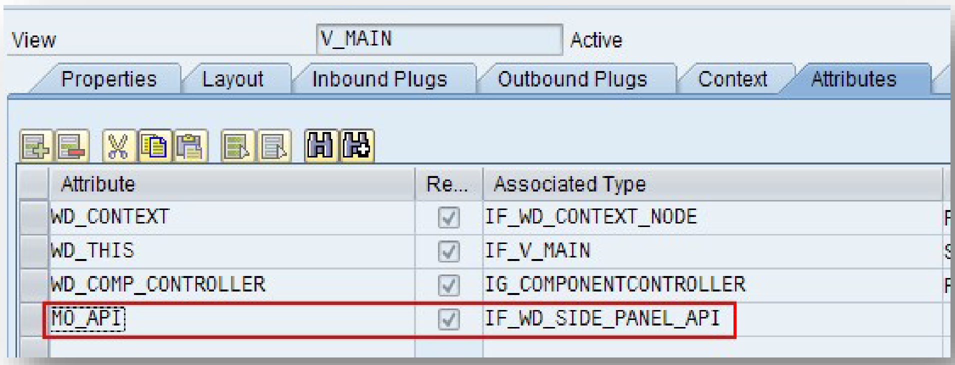
1. Create an empty view. This view will contain the coding to integrate BCV as described below.
2. Place an empty UI container in this view.
3. Integrate the POWL view to this UI container.

   We do not recommend using this implementation model, but it is currently the only option to integrate BCV.

Note that adding the header to the view does not duplicate the header in the Business Client.

2.2. Side Panel Initialization

Add the attribute MO_API (type reference to IF_WD_SIDE_PANEL_API) to the attributes of your view:



Attribute	Re...	Associated Type
WD_CONTEXT	✓	IF_WD_CONTEXT_NODE
WD_THIS	✓	IF_V_MAIN
WD_COMP_CONTROLLER	✓	IG_COMPONENTCONTROLLER
MO_API	✓	IF_WD_SIDE_PANEL_API

Add the attribute MO_API (type reference to IF_WD_SIDE_PANEL_API) to the attributes of your view:

```
*---------------------------------------------------------------------*
*    This coding checks if BCV is active and if the user has the
*    authority to execute the side panel. If the user is allowed to do
*    this, the side panel is initialized.
*---------------------------------------------------------------------*
     CONSTANTS:
*    The context key „/PLMU/MATERIAL“ is just an example, please adapt
*    it to your needs
     lc_context_key     TYPE /bcv/fnd_context_key VALUE '/PLMU/MATERIAL'.

     DATA:
     lv_authorized      TYPE boole_d,
     lo_view           TYPE REF TO if_wd_view.

*     Check if BCV is active. If it is not active the side panel can't be used
     IF /bcv/cl_fnd_bcv_switch_check=>bcv_img_01( ) = abap_false.
       RETURN.
     ENDIF.

*     Check if the user has the authority to execute the side panel
     lv_authorized = /bcv/cl_uif_ext_assist=>check_execute_sidepanel(
       iv_context_key = lc_context_key
       iv_uname       = sy-uname ).

     IF lv_authorized = abap_false.
*     Authority for side panel execution not granted
       RETURN.
     ENDIF.

     lo_view ?= wd_this->wd_get_api( ).

*     Get Side Panel API
     wd_this->mo_api = cl_wd_side_panel_api=>get_api( ).

*     Initialize Side Panel
     wd_this->mo_api->init(
       EXPORTING
               side_panel_config_id    = '/BCV/SIDEPANEL'
               *  apply_configuration  = ABAP_TRUE
               is_open                 = abap_true
               allow_side_panel_config = abap_true
               view_controller         = lo_view
               open_action_name        = 'OPEN_SP'
               auto_refresh            = abap_true
               *    width               = 410
               link_text               = 'Side Panel'
               link_text_config_mode   = 'Side Panel Config Mode' ).
```

For a full parameter explanation, see the interface of the API. The following parameters are required to display the BCV side panel correctly:

Mandatory Parameters

- The parameter **SIDE_PANEL_CONFIG_ID** determines which side panel is displayed. For the BCV side panel, the correct value is '/BCV/SIDEPANEL' (EhP5) or '/BCV/SIDEPANEL_20' (EhP6) only.
- The parameter **OPEN_ACTION_NAME** describes the action you have to define in your view (in the example above, it is 'OPEN_SP'). This action is used to open the side panel.

Optional Parameters

- The parameter **IS_OPEN** defines whether the side panel opens automatically at the start of the hosting application or if it has to be opened manually by the user.
- The parameter **WIDTH** describes the initial width of the BCV side panel. BCV side panel content is optimized for screen resolution 1280x1024. Do not set this parameter as the side panel gets its default width from the BCV configuration.

2.2.1. Web Dynpro Action

You must also define an action for your view (for example, 'OPEN_SP'). See section *Side Panel Initialization*. The action is triggered using the Web Dynpro ABAP Page Builder to open the side panel when the user chooses the *Side Panel* link.

The *Side Panel* link is displayed in the upper right-hand corner of the screen in the page header.

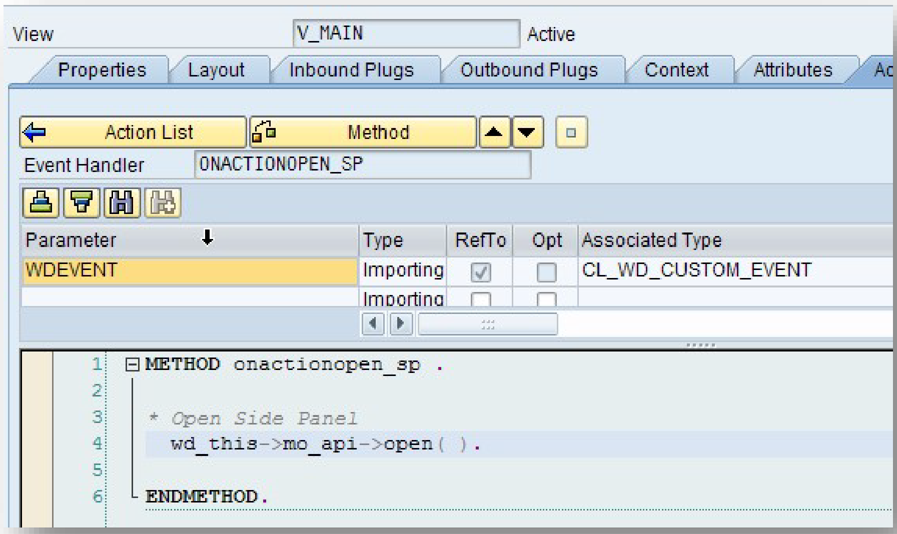
When using this integration in NetWeaver EhP2, there is currently no option for changing or overwriting the link *Additional Information*.

The action OPEN_SP to open the side panel was already registered in the previous step (see the coding extract above) and will be triggered when the user chooses the *Side Panel* link on the UI.

```
* Open Side Panel
wd_this->mo_api->open( ).
```

When initializing the side panel, the API reference (mo_api) is kept on Web Dynpro view level for reuse.

When initializing the side panel, the API reference (mo_api) is kept on Web Dynpro view level for reuse.



Event Handler ONACTIONOPEN_SP

Parameter	Type	RefTo	Opt	Associated Type
WDEVENT	Importing	✓		CL_WD_CUSTOM_EVENT
	Importing			

```
METHOD onactionopen_sp .

  * Open Side Panel
    wd_this->mo_api->open( ).

ENDMETHOD.
```

Example: Implementation of the action OPEN_SP

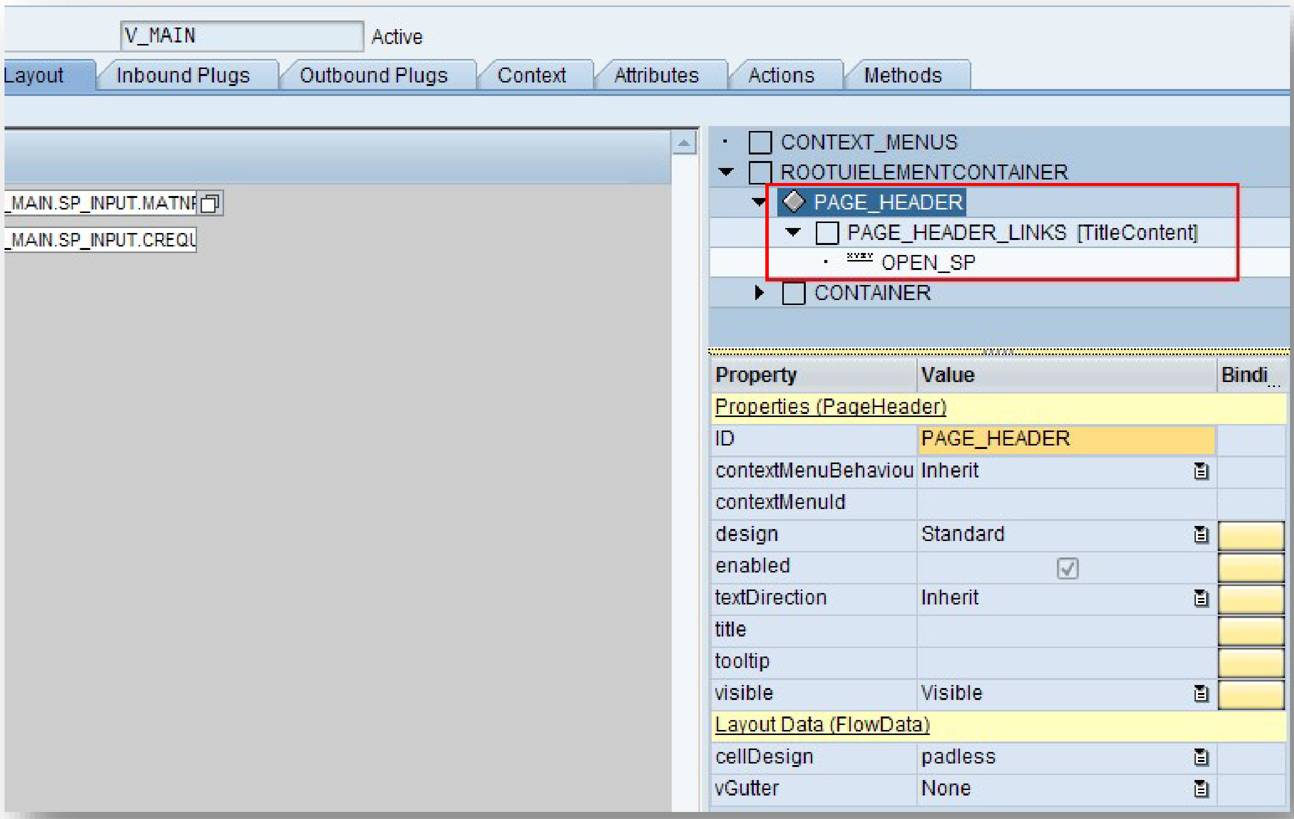
2.2.2. Web Dynpro UI Element Page Header

In order to display the *Side Panel* link on the UI, ensure that you have a page header in your application. The page header ID should be named PAGE_HEADER. The Page Builder requires this naming to reach the correct target required to create the *Side Panel* link.

Test the implementation. You should be able to open and close the side panel, which is usually empty at this point, with container items for *Overview*, *Query Views*, *Dashboards*, *Favorites*, and *Launchpad*.

If you are getting error messages, enhance the view in the following way:

- Add a “transparent container” (title content) to the page header with the ID PAGE_HEADER_LINKS.
- Add a “link to action” to the transparent container with the ID OPEN_SP. The link has to reference the action created in the Web Dynpro action. See section *Web Dynpro Action*.



Property	Value	Bindi...
Properties (PageHeader)		
ID	PAGE_HEADER	
contextMenuBehaviou	Inherit	
contextMenuId		
design	Standard	
enabled	✓	
textDirection	Inherit	
title		
tooltip		
visible	Visible	
Layout Data (FlowData)		
cellDesign	padless	
vGutter	None	

Example: Layout definition of view

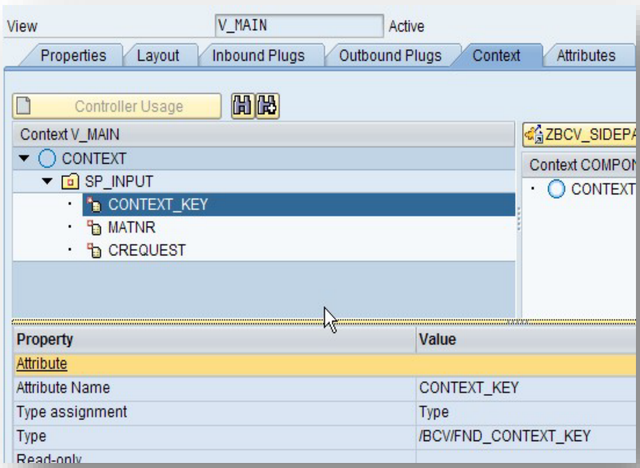
2.3. Side Panel Update (Tagging)

After you have initialized the side panel, you must update it with information about your application. This update is called tagging. It can be done using code or configuration. However, we recommend that at least the context key (an ID defined in BCV Customizing to identify your application) should be tagged using code as it might not be available on the UI for configuration. In plain Web Dynpro ABAP, you use the NetWeaver tagging service for tagging using code.

Tagging by configuration works in the same way for all the different approaches and is therefore explained at the end of this document.

2.3.1. Tagging Service

The tagging service can be accessed using class CL_WD_TAGGING_MANAGER. You can create a connection between your application data and the side panel using the methods listed below. In the example below, it is presumed that you have defined a Web Dynpro context node named SP_INPUT with several attributes, for example, an attribute called CONTEXT_KEY and another attribute called MATNR.



Property	Value
Attribute	
Attribute Name	CONTEXT_KEY
Type assignment	Type
Type	/BCV/FND_CONTEXT_KEY
Read-only	

Example: Definition of Web Dynpro context

The actual values of the context attributes in the example above can be transferred to the side panel using the coding example below. This type of coding could be implemented in the WDDOINIT method of your Web Dynpro view.

```
*--------------------------------------------------------------------*
* This coding is using the tagging service to do the tagging via
* coding. The tags are bounded to the attributes of the view context.
*--------------------------------------------------------------------*
  DATA:
    lo_tagging_service TYPE REF TO if_wd_tagging_service.

* Get instance of tagging service
  lo_tagging_service = cl_wd_tagging_manager=>get_service_by_component(
       component = wd_comp_controller->wd_get_api( ) ).

* Tag the context key with the actual value of the context attribute
* CONTEXT_KEY. Pay attention: tag name for context key is always
* „/BCV/:CONTEXT_KEY"
  lo_tagging_service->set_tag_by_path_context(     EXPORTING
      context_path = 'SP_INPUT.CONTEXT_KEY'
      wd_context   = wd_context
      tag          = '/BCV/:CONTEXT_KEY'
      scope        = 00 ).

* Tag the meaning 1MATERIAL with the actual value of the context attribute
* MATNR. Pay attention: tag name is the concatenation of „/BCV/:" and
* the name of the meaning. This means you have to adapt this coding to
* your needs.
  lo_tagging_service->set_tag_by_path_context(
  EXPORTING
      context_path    =    'SP_INPUT.MATNR'
      wd_context    =           wd_context
```

```
        tag              = '/BCV/:1MATERIAL'
        scope          = 00 ).
* Tag the system alias in case your application runs in SAP Portal. See
* also the chapter "System Alias" for further information
  lo_tagging_service->set_tag_by_path_context(
  EXPORTING
      context_path = 'SP_INPUT.SYSALIAS'
      wd_context   = wd_context
      tag          = '/BCV/:SYSALIAS'
      scope        = 00 ).
* Tag additional meanings here: ...
```

This coding example transfers the **mandatory BCV tag** CONTEXT_KEY with your current context key value to the side panel. This key is used to determine the content of the BCV side panel. Only BCV query views assigned to this context key are available in BCV in the current session. You can use the **data element /BCV/FND_CONTEXT_KEY** to define the Web Dynpro context attributes containing the context key as constant, for example. See section *Context Key*.

Furthermore, in the example above, the BCV meaning 1MATERIAL is connected for data value transfer. You can easily extend your data transfer by using the SET_TAG_BY_PATH_CONTEXT method of the tagging service.

When working with tags, consider the following:

- Tags from the database table /BCV/C_MEAN must use the leading prefix `/BCV/:`
System tags such as /BCV/:CONTEXT_KEY and /BCV/:SYSALIAS constants can be reused from class **/BCV/CL_UIF_EXT_ASSIST constants** /BCV/TAG_CONTEXT_KEY and /BCV/TAG_SYSALIAS.
- There is no restriction to the number of tags you can use.
- Any attribute can have several tags.
- You cannot use the same tag for more than one attribute on the same screen, as this causes corruption in the tagging service.
- Only one value can be transferred for each field or tag, for example, the transfer of a table column is not supported. If you try to tag a table column, only the value of the current line is transferred.

The tagging service methods use the following parameters:

- "**context_path**" describes the path where your Web Dynpro context attribute is stored, built out of a node and attribute.

  In the above example, the node is SP_INPUT and the attribute is MATNR.
- "**wd_context**" refers to your local component context, containing the context node.
- "**tag**" describes the name of your tag.
Note that the name of your corresponding BCV meaning should be inserted. See section *Tag vs. Meaning*.
- **"scope"** describes in which scope your tagging was set. Value 00 is used for coding-defined tagging (as standard). Value 04 is reserved for personalization and should not be used in this context.

3. Floorplan Manager Applications

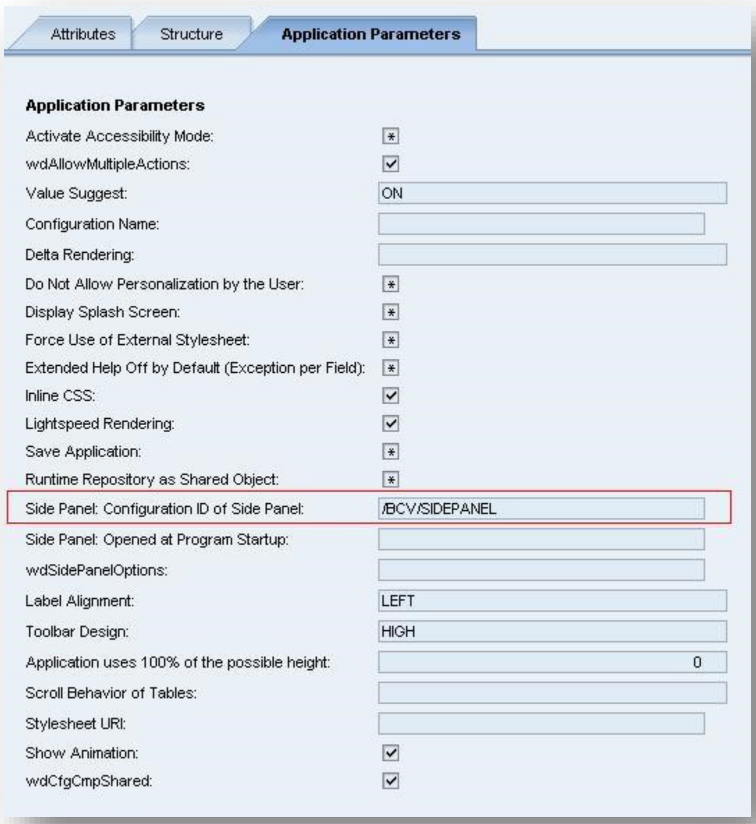
For all FPM-based approaches, the side panel is defined in the same way, using the *Configuration ID of Side Panel* application parameter in your FPM application configuration.

3.1. WebDynpro Application Configuration

To access the Web Dynpro application configuration of your FPM-based Web Dynpro application, go to transaction SE80 and follow one of the paths below:

- */nSE80 → Web Dynpro Component „FPM_OIF_COMPONENT" → Web Dynpro Applications → <your application> → Create/Change Configuration (context menu at the application entry) → <select your Configuration ID> → Buttons „Change"/"Display" → tab strip „Application Parameters"*
- */nSE80 → <your package> → Web Dynpro → Application Configurations → <select your configuration> → Button „Start Configurator" → Buttons „Change"/"Display" → tab strip „Application Parameters"*

To enable the BCV side panel, choose the *Application Parameters* tab and insert the value `/BCV/SIDEPANEL` (EhP5) or `/BCV/SIDEPANEL_20` (EhP6) into the attribute *Side Panel: Configuration ID of Side Panel*.



Application Parameters

Activate Accessibility Mode:	*
wdAllowMultipleActions:	✓
Value Suggest:	ON
Configuration Name:	
Delta Rendering:	
Do Not Allow Personalization by the User:	*
Display Splash Screen:	*
Force Use of External Stylesheet:	*
Extended Help Off by Default (Exception per Field):	*
Inline CSS:	✓
Lightspeed Rendering:	✓
Save Application:	*
Runtime Repository as Shared Object:	*
Side Panel: Configuration ID of Side Panel:	/BCV/SIDEPANEL
Side Panel: Opened at Program Startup:	
wdSidePanelOptions:	
Label Alignment:	LEFT
Toolbar Design:	HIGH
Application uses 100% of the possible height:	0
Scroll Behavior of Tables:	
Stylesheet URI:	
Show Animation:	✓
wdCfgCmpShared:	✓

Example: Defining the side panel in application configuration

Several additional side panel and application parameter options are available in FPM also. To recheck the functions, refer to the described parameter set in section *"Plain" Web Dynpro ABAP Application*.

3.2. FPM Application Based on PLM UI Framework

If you have an FPM application that is based on the PLM UI Framework, you must perform the following steps:

- Application configuration (see section *Web Dynpro Application Configuration*)
- Side panel initialization (or authority check)
- Side panel update (tagging)

3.2.1. Side Panel Initialization

The initialization and tagging of the side panel for the PLM UI framework is done by implementing the BAdI /PLMU/EX_FRW_SIDEPANEL. This BAdI provides the interface that you have to implement, /PLMU/IF_EX_FRW_SIDEPANEL. In your BAdI implementation, you can use the following code to perform the "initialization", essentially an authority check that allows you to switch off the side panel if the BCV business function is not active or if the user does not have the corresponding authority.

```
METHOD /plmu/if_ex_frw_sidepanel~check_authority.
*---------------------------------------------------------------------*
* This method checks if the user has the authority to execute the
* BCV side panel.
* <-> CV_FAILED is ABAP_FALSE if the user has the authority, otherwise
*      it is ABAP_TRUE.
*---------------------------------------------------------------------*
   CONSTANTS:
*  The context key „/PLMU/MATERIAL" is just an example, please adapt
*  it to your needs.
   lc_context_key TYPE /bcv/fnd_context_key VALUE '/PLMU/MATERIAL'.

   DATA:
         lv_authorized  TYPE boole_d.

*  If the authorization check already failed in another BAdI implementation,
*  skip this one.
   IF cv_failed = abap_true.
       RETURN.
   ENDIF.

*  Check if BCV is active. If it is not active the side panel can't be used.
   IF /bcv/cl_fnd_bcv_switch_check=>bcv_img_01( ) = abap_false.
       RETURN.
   ENDIF.

*  Check if the user has the authority to execute the side panel
*  with your context key
   lv_authorized = /bcv/cl_uif_ext_assist=>check_execute_sidepanel(
       iv_context_key = lc_context_key
       iv_uname       = sy-uname ).

    IF lv_authorized = abap_true.
    *      Grant authority for side panel execution
```

```
        CLEAR cv_failed.
    ENDIF.

ENDMETHOD.
```

3.2.2. Side Panel Update (Tagging)

The tagging is also done by implementing the BAdI /PLMU/EX_FRW_SIDEPANEL. Refer to the example implementation below. The parameter used, GV_MATNR, is just a global attribute of the BAdI implementation class. The attribute must be set by your application coding.

```
METHOD /plmu/if_ex_frw_sidepanel~get_param_tag_map.
*--------------------------------------------------------------------*
* This method does the tagging of the BCV side panel. This means that
* the context key and meanings are transferred from the hosting
* application to the side panel to start it with appropriate values.
* Pay attention to the naming conventions of the BCV tags, they are
*  beginning with '/BCV/:'.
*--------------------------------------------------------------------*
   DATA:
      ls_tag             TYPE mdg_bs_mat_s_mp_mat_tags,
      ls_param_tag_map   LIKE LINE OF ct_param_tag_map.

*   Do not clear the parameters as they might have been changed already in
*   another BAdI implementation.
*   CLEAR:
*      cs_no_sidepanel_message,
*     ct_param_tag_map.

*   Tag the context key with the value of your context key (/PLMU/MATERIAL in
*   this example). Pay attention: tag name for context key is always
*   „/BCV/:CONTEXT_KEY"
   ls_param_tag_map-name  = 'CONTEXT_KEY'.
   ls_param_tag_map-tag   = '/BCV/:CONTEXT_KEY'.
   ls_param_tag_map-value  = '/PLMU/MATERIAL'.      "Use your own context key here
   INSERT ls_param_tag_map INTO TABLE ct_param_tag_map.

*    Tag  the  meaning  1MATERIAL  with  the  actual  value  of  the  global  attribute
*   GV_MATNR. Pay attention: tag name is the concatenation of „/BCV/:" and
*    the     name     of     the     meaning     (in     this     case     meaning     1MATERIAL).
*   This means you have to adapt this coding to your needs.
  IF gv_matnr IS NOT INITIAL.
      CLEAR ls_param_tag_map.
      ls_param_tag_map-name   = 'MATNR'.
      ls_param_tag_map-tag    = '/BCV/:1MATERIAL'.
      ls_param_tag_map-value   = gv_matnr.
      INSERT ls_param_tag_map INTO TABLE ct_param_tag_map.
   ENDIF.
*   Tag the system alias in case your application runs in SAP Portal. Please see
*   also the chapter "System Alias" for further information
    IF gv_sysalias IS NOT INITIAL.
      CLEAR ls_param_tag_map.
      ls_param_tag_map-name   = 'SYSALIAS'.
      ls_param_tag_map-tag    = '/BCV/:SYSALIAS'.
      ls_param_tag_map-value   = gv_matnr.
      INSERT ls_param_tag_map INTO TABLE ct_param_tag_map.
    ENDIF.
ENDMETHOD.
```

As mentioned above, you can also tag meanings by configuration. See section *Tagging using Configuration*.

To **trigger a refresh of the side panel**, for example, because an attribute of your application such as the material number has changed, you can use method

/PLMU/CL_FRW_APPL_CNTRL=>REFRESH_SIDEPANEL( ).

3.3. FPM Overview Page Application using Generic User Interface Building Blocks

If you have an FPM application that is not based on the PLM UI framework but uses the Overview Page (OVP) floor plan and GUIBBs, you have to perform the following steps:

- Application configuration (see **3.1 Web Dynpro Application Configuration**)
- Side panel initialization
- Side panel update (tagging)

3.3.1. Side Panel Initialization

To initialize the side panel, you must set the side panel link in the context navigation region (CNR) of the OVP floor plan. You can do this in your coding when you have access to an FPM instance, for example, in the application controller or in a feeder class.

```
*---------------------------------------------------------------------*
* The following coding checks if the user has the authority to
*   execute the BCV side panel.
*---------------------------------------------------------------------*

  CONSTANTS:
*   The context key „/PLMU/MATERIAL" is just an example, please adapt
*   it to your needs.
    lc_context_key TYPE /bcv/fnd_context_key VALUE '/PLMU/MATERIAL'.

  DATA:
      lv_authorized  TYPE boole_d,
      lo_fpm    TYPE REF TO if_fpm,
      lo_cnr_ovp     TYPE REF TO if_fpm_cnr_ovp.

*   Check if BCV is active. If it is not active the side panel can't be
*   used and should not be visible.
  IF /bcv/cl_fnd_bcv_switch_check=>bcv_img_01( ) = abap_false.
      RETURN.
  ENDIF.

* Check if the user has the authority to execute the side panel   lv_authorized
  = /bcv/cl_uif_ext_assist=>check_execute_sidepanel(
      iv_context_key = lc_context_key
      iv_uname       = sy-uname ).

  IF lv_authorized = abap_false.
*   Authority for side panel execution not granted
    RETURN.
  ENDIF.
```

```
*     Set Side Panel link
  lo_fpm = cl_fpm_factory=>get_instance( ).
  lo_cnr_ovp ?= lo_fpm->get_service(cl_fpm_service_manager=>gc_key_cnr_ovp ).
  lo_cnr_ovp-set_side_panel_link(
    iv_text    = 'Side Panel'
    iv_tooltip = 'Open Side Panel'
    iv_active  = abap_true ).

*   Set tagging value for context key
  lo_cnr_ovp->set_tag_value(
    iv_tag   = '/BCV/:CONTEXT_KEY'
    i_value = lc_context_key ).

* Set tagging value for system alias
  lo_cnr_ovp->set_tag_value(
    iv_tag   = '/BCV/:SYSALIAS'
    i_value = lv_sysalias ). "Please see chapter "System Alias"
```

If you want to change the value of a tag (here: context key) during the runtime of your application, you should make sure that all values passed to the I_VALUE parameter of method SET_TAG_VALUE for this tag have the *same* data type. Otherwise, subsequent calls of SET_TAG_VALUE might not have an effect, meaning the value change might be ignored. The easiest way to ensure the same data type is to use a constant or variable with this data type to pass the tag value to I_VALUE (instead of using a *literal* like `'MY_CONTEXT_KEY'` directly).

3.3.2. Side Panel Update (Tagging)

The tagging of parameters for your application should be done using configuration. For more information, see section *Tagging using Configuration*.

3.4. FPM Application using Freestyle User Interface Building Blocks

If you have an FPM application that is not based on the PLM UI framework or using the OVP floor plan, you must perform the following steps:

- Application configuration (see section *Web Dynpro Application Configuration*)
- Side panel initialization
- Side panel update (tagging)

Precondition

Your FPM application should have at least one freestyle UIBB. If your FPM application does not yet have a freestyle UIBB, you can create a new Web Dynpro view with no UI elements and use this as a freestyle UIBB in your application. In the Web Dynpro view of this freestyle UIBB, you can place the coding used to initialize the side panel.

3.4.1. Side Panel Initialization

Add the following coding to your freestyle UIBB, for example, to the WDDOINIT method of your view.

```
*----------------------------------------------------------------------*
* The following coding checks if the user has the authority to
*  execute the BCV side panel and sets the side panel link.
*----------------------------------------------------------------------*
  CONSTANTS:
*  The context key „/PLMU/MATERIAL“ is just an example, please adapt
*  it to your needs.

    lc_context_key TYPE /bcv/fnd_context_key VALUE '/PLMU/MATERIAL'.

  DATA:
    lv_authorized  TYPE boole_d,
    lo_idr         TYPE REF TO if_fpm_idr.

*      Check if BCV is active. If it is not active the side panel can't be
*      used and should not be visible.

  IF /bcv/cl_fnd_bcv_switch_check=>bcv_img_01( ) = abap_false.
    RETURN.
  ENDIF.

*      Check if the user has the authority to execute the side panel
   lv_authorized = /bcv/cl_uif_ext_assist=>check_execute_sidepanel(
     iv_context_key = lc_context_key
     iv_uname       = sy-uname ).

  IF lv_authorized = abap_false.
*      Authority for side panel execution not granted
    RETURN.
  ENDIF.

*    Get IDR service
   lo_idr ?= cl_fpm_service_manager=>get_service(
     cl_fpm_service_manager=>gc_key_idr ).

*      Initialize side panel and set side panel link
   lo_idr->set_side_panel_link(
     iv_text    = 'Side Panel'
     iv_tooltip = 'Open Side Panel'
     iv_active  = abap_true ).

*      Set tagging value for context key
   lo_idr->set_tag_value(
     iv_tag   = '/BCV/:CONTEXT_KEY'
     i_value = lc_context_key ).
*      Set tagging value for context key
   lo_idr->set_tag_value(
     iv_tag   = '/BCV/:SYSALIAS'
     i_value = lv_sysalias ). "Please see chapter “System Alias”
```

If you want to change the value of a tag (here: context key) during the runtime of your application, you should make sure that all values passed to the I_VALUE parameter of method SET_TAG_VALUE for this tag have the *same* data type. Otherwise, subsequent calls of SET_TAG_VALUE might not have an effect, meaning the value change might be ignored. The easiest way to ensure the same data type is to use a constant or variable with this data type to pass the tag value to I_VALUE (instead of using a *literal* like `'MY_CONTEXT_KEY'` directly).

The code can also be placed in the WDDOINIT method of the window containing the views.

3.4.1. Side Panel Update (Tagging)

The tagging of parameters of your application should be done using configuration. For more information, see section *Tagging using Configuration*.

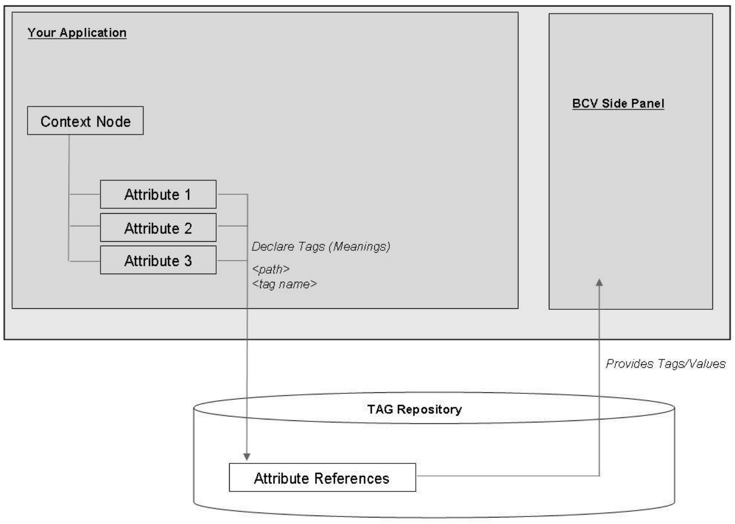
4. Data Transfer

To get a context-sensitive side panel that adapts to the currently displayed business object of the hosting application, data has to be transferred from the application to the side panel.

The data identifies the hosting application (using the context key) and the currently displayed object, for example, the ID of the product or customer.

To transfer data from your application to the side panel and on to further BCV pages, you use the NetWeaver tagging service. For data transfer to the BCV side panel, you require a Web Dynpro context node in your application. The context node contains the attributes which are connected to the tagging service.

Setting up the connection between the context node attributes and tags is the first step. This defines the path of the relation. The second step of data transfer takes place automatically when attributes are filled or changed.



All context attributes with tags are transferred to the side panel.

Each time a context attribute value changes, the tagging service reports the value change to the side panel. This is the default setting. The API of the Page Builder provides methods to disable the automatic transfer of the tagged attributes. If automatic transfer is disabled, the hosting application must trigger the transfer, or enable the automatic transfer of the tagged attributes again.

4.1. Tagging Service

4.1.1. Tag vs. Meaning

Tags are used as application-independent virtual field names in the same way as tags used in HTML documents. The term “tag” is used in the context of the Page Builder.

For BCV, this function is taken over by the meaning. The meanings are independent from the technical field names used by the different data providers, for example, BI or ES.

Values with the same meaning are related and can be compared without any additional conversion. They are based on the same definition and belong to the same value range.

> Example:
>
> It is assumed that two fields with the tag 1MATERIAL contain material IDs. If the values are equal, they are identifying the same object.

The tags you use in the following code section must be used as meanings in BCV. These meanings are assigned to the fields of BCV queries. Values with matching meanings are transferred as input parameters to the BCV queries and used to receive data from different data providers.

If the tags do not exist as meanings in BCV, you won’t get an error message, but the data is not selected in the way that you are expecting.

The meanings depend on your application and the scenarios the customer is using.

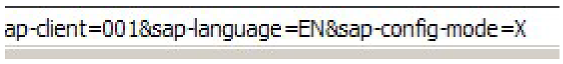
The only mandatory tag is the `CONTEXT_KEY`.

4.1.2. Tagging using Configuration

The tags can also be assigned using configuration. This configuration can enhance or overwrite the tagging assignment done in the code.

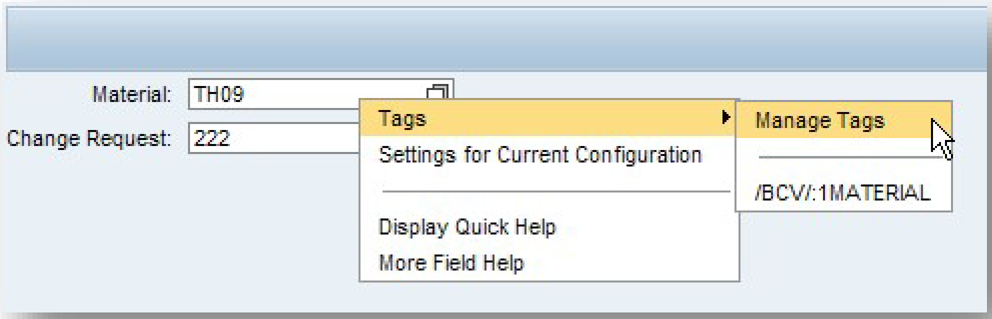
Customers can use this feature to adapt the tagging to their needs without modifying the hosting application. Perform the following steps:

1. Call your application and open the Web Dynpro screen that contains the *Side Panel* link.
2. Add “&sap-config-mode=x” to the URL in the browser and press Enter.

   ap-client=001&sap-language=EN&sap-config-mode=X

   Example: URL enhanced with sap config mode
3. Right mouse click on the field that should be tagged and choose *Tags* → *Manage Tags*.



4. In the dialog box that appears, you can add your own tags or remove existing ones. As mentioned before, you can assign several tags to the same field or attribute.

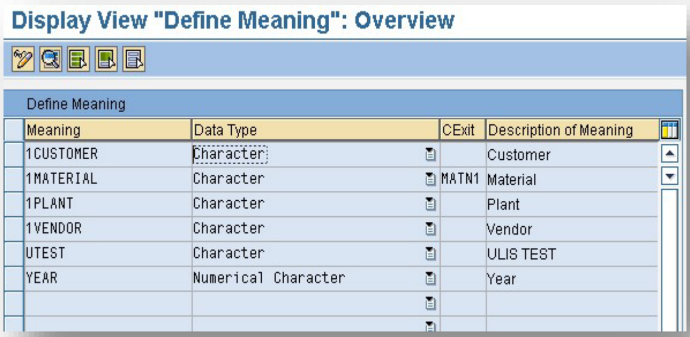
4.1.3. Example Data Transfer

Here is an example of data transfer with tags and meanings:

1. You have defined a BCV query and query view that require the application parameters material and customer in order to display an analysis of materials bought by different customers.

   The values of both parameters are available in your Web Dynpro application.

2. Define the BCV meanings to represent material and customer, for example, 1MATERIAL and 1CUSTOMER:

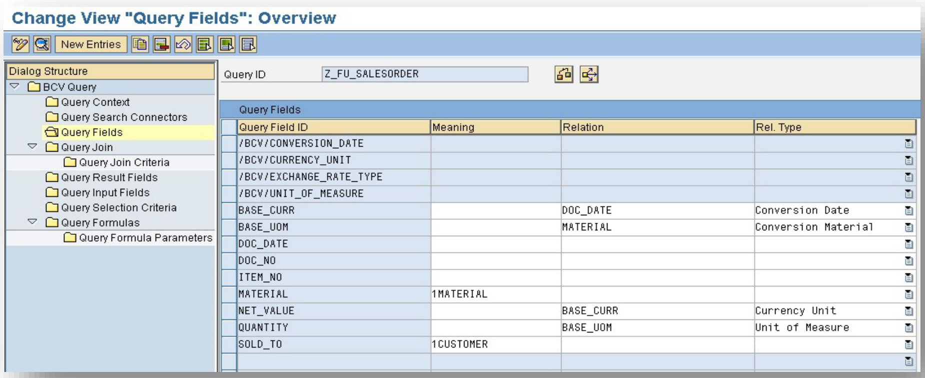


Display View "Define Meaning": Overview

Define Meaning

Meaning	Data Type	CExit	Description of Meaning
1CUSTOMER	Character		Customer
1MATERIAL	Character	MATN1	Material
1PLANT	Character		Plant
1VENDOR	Character		Vendor
UTEST	Character		ULIS TEST
YEAR	Numerical Character		Year

3. Assign the meanings to your query fields and query view input parameters in BCV Customizing or in the configuration center:


Change View "Query Fields": Overview

Query ID Z_FU_SALESORDER

Query Fields

Query Field ID	Meaning	Relation	Rel. Type
/BCV/CONVERSION_DATE			
/BCV/CURRENCY_UNIT			
/BCV/EXCHANGE_RATE_TYPE			
/BCV/UNIT_OF_MEASURE			
BASE_CURR		DOC_DATE	Conversion Date
BASE_UOM		MATERIAL	Conversion Material
DOC_DATE			
DOC_NO			
ITEM_NO			
MATERIAL	1MATERIAL		
NET_VALUE		BASE_CURR	Currency Unit
QUANTITY		BASE_UOM	Unit of Measure
SOLD_TO	1CUSTOMER		

4. Set up the tagging service as described above to transfer the values (using tags or meanings) from your application to BCV, and thus to your query views. Connect material and customer (from your Web Dynpro context node attributes) with BCV using the method SET_TAG* by using the BCV meaning as a tag name. See 4.1.3 on **Example Data Transfer**.

   This also includes the tagging of the context attribute containing the context key. Note that the tag CONTEXT_KEY does not exist in the meaning table and is a “fix” or “virtual” tag or meaning, always required when using BCV (so do not search for it in this table).

Important

1. The tagging service is Web-Dynpro-runtime-model sensitive.
   This means you must start the tagging before the navigation phase of Web Dynpro, ideally on WDDOINIT.
2. You should only set a tag once.
   Do not set a tag or name more than once in a session. Otherwise, the tagging service becomes corrupt and tags are not available.

4.2. Context Key

The BCV context key is the **identifier** of the hosting application. It defines the available BCV queries, query views (CHIPs), and side panel content.

Using the context key applications, you can create different side panel content for different object types. The key avoids a mix-up of BCV objects within one system, as the validity of BCV objects is checked for the current context key.

The first usage of context key occurs when opening the side panel. Applications using BCV have to transfer their context key to the side panel. The BCV side panel will only display consistent content when a valid context key is used.

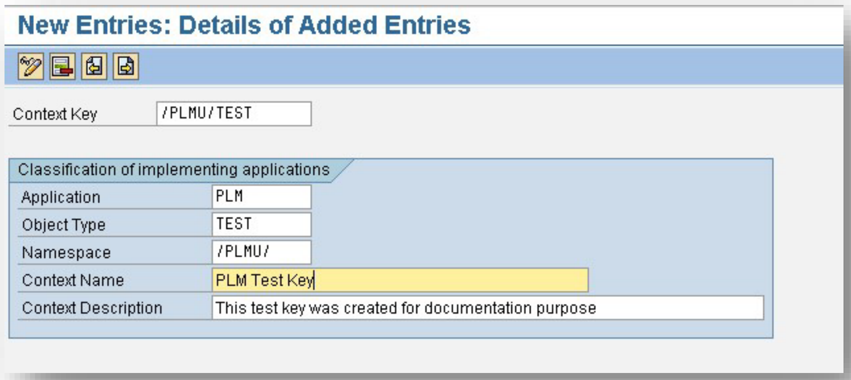
If you are using BCV with your application, you can create your own context key. When creating context keys for your application, refer to the table below.

The BCV context key is used in the following BCV objects:

BCV Object	Context Key Relation	Other Relations
Query	Queries can be assigned to more than one context key for which they are valid	BCV Customizing
Query View	Query views get their assignment or validity through dependency to the underlying query	BCV Customizing (and dependency to parent query)
Dashboard	Context key is part of the object key. Only query views assigned to that context key can be inserted into the dashboard.	The head is created in BCV Customizing. Layout and content are stored on the NetWeaver Web Dynpro configuration side
Overview	Overview list shown on side panel start-up. It has the same behavior as the dashboard.	The same relations as in dashboards
Grouping	Hierarchical list of query views for fast access available in side panel. The context key is also part of the object key.	BCV Customizing
Side Panel	Includes all the above BCV objects. Side panel content is retrieved for only one context key and thus represents the current compilation of objects within this key area.	The content is determined by BCV objects that are valid for the current context key.

4.2.1. Creation of BCV Context Key

To create a context key, navigate to the view /BCV/V_CLF using view maintenance (transaction SM30) and create a new entry for your application.


New Entries: Details of Added Entries

Context Key	/PLMU/TEST

Classification of implementing applications

Application	PLM
Object Type	TEST
Namespace	/PLMU/
Context Name	PLM Test Key
Context Description	This test key was created for documentation purpose

Example: View Maintenance for BCV Context Keys

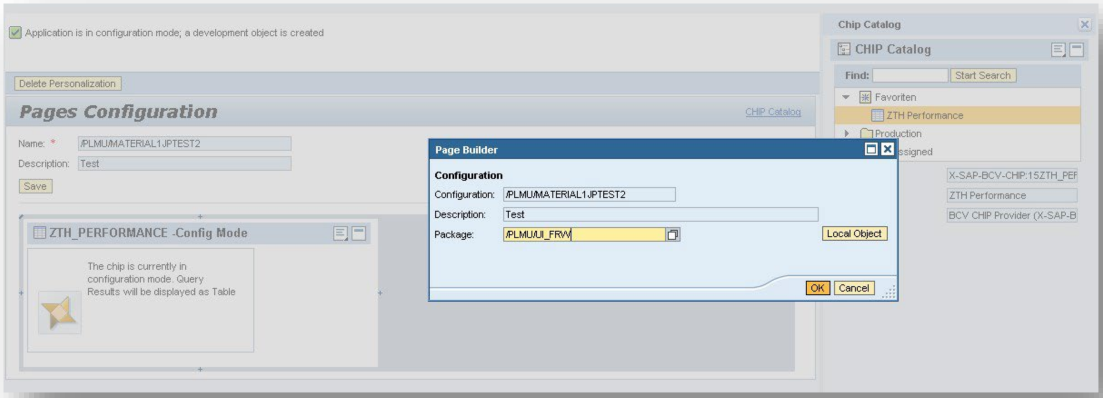
For the context key itself, choose a key using the following criteria:

- It represents your application
- If you are using namespaces, it includes a valid namespace of your area
- It has a “sub” description (for example, business object) in the key in case you want to create more than one context key

Context Key and Namespace

Note that when you are using namespaces in your development component, you must ensure that your context key begins with a valid namespace. This is extremely important for later BCV Customizing. BCV uses NetWeaver Web Dynpro ABAP Page Builder technology for the dashboard, overview, and side panel creation. NetWeaver builds up these BCV dashboards and overviews using Web Dynpro configurations. The ID of such a configuration is prepared by BCV and represents a concatenation of [NAMESPACE], [CONTEXT_KEY] and [DASHBOARD/OVERVIEW_ID].

The creation is connected to a transport request mechanism and requires you to save a valid package. As the name of this Web Dynpro configuration object will begin with the current context key (and thus your namespace), choose a valid namespace of your application in which those development objects will be stored.



Application is in configuration mode; a development object is created

Name: * /PLMU/MATERIAL1JPTEST2
Description: Test

The chip is currently in configuration mode. Query Results will be displayed as Table

Configuration: /PLMU/MATERIAL1JPTEST2
Description: Test
Package: /PLMU/UI_FRW

X-SAP-BCV-CHIP:15ZTH_PEF
ZTH Performance
BCV CHIP Provider (X-SAP-B

When using the full length of namespace (10 characters), there will be 6 characters left for the specification of the object type.

Example:

- In the PLM example above, the context key is built from the namespace /PLMU/ (which also identifies the hosting application) plus the identification of the business object material.
- Once the PLM connects other object types to the BCV side panel, /PLMU/BOM is created for bill of material or /PLMU/DOCUMENT for documents.

Other Attributes

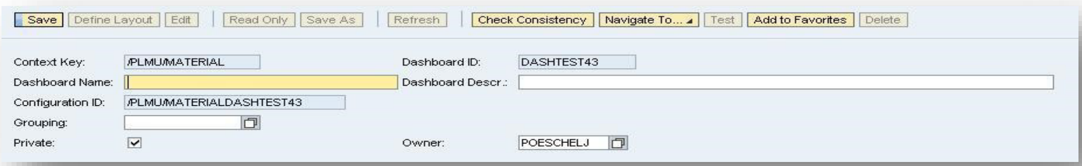
Further attributes such as **object type**, **name,** and **description** are generally used for administrative purposes.

4.2.1.Namespace of Context Key vs. Configuration ID

When your application uses a namespace, you should make the namespace part of your context key. The technical background is the usage of the Page Builder function when creating BCV dashboards and overviews.

The connection between the BCV objects dashboard and overview and the NetWeaver pages is realized in the database table attribute CONFIG_ID of both BCV database tables. The configuration ID is then used on the NetWeaver side to store corresponding pages as Web Dynpro configurations (component name: WDR_CHIP_PAGE).

When creating a dashboard or overview, **BCV provides you with a suggestion for a configuration ID**. You can decide to use that suggestion or insert your own key. However, when your context key starts with a valid namespace, we recommend that you use the suggestion as it exactly describes the used BCV object on the NetWeaver side (→ key concatenation).



Context Key:	/PLMU/MATERIAL	Dashboard ID:	DASHTEST43
Dashboard Name:		Dashboard Descr.:	
Configuration ID:	/PLMU/MATERIALDASHTEST43		
Grouping:			
Private:	✓	Owner:	POESCHELJ

Example: Proposal of Configuration ID – concatenation of BCV Context Key and Object ID

Note

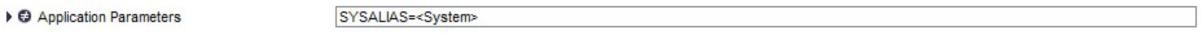
Note that you cannot change the ID in the configuration center or maintenance view afterwards.

When maintaining dashboards or overviews using Customizing or view maintenance, the configuration ID is created in the background (using concatenation). If the ID already exists, you will be asked to create your object in the BCV configuration center (for configuration ID maintenance reasons).

4.3. System Alias

If the application that hosts the *Side Panel* link runs in a portal environment also, you must provide the system alias for navigation within the portal. The system alias is usually provided in your iView. To get the alias during Web Dynpro runtime, use the following retrieval process:

Add an application parameter to your iView: SYSALIAS=<System>. <System> is the command for the iView to pass the value of current system alias. The URL parameter SYSALIAS is the container that can be accessed on the Web Dynpro application side, retrieving the specified system alias.



Application Parameters	SYSALIAS=<System>

Example: iView Detail

Within an FPM-based application, you can retrieve the system alias as in the following example:

```
DATA: lr_parameter         TYPE REF TO if_fpm_parameter,
       lv_value           TYPE ihttpval.

*      MO_FPM / Instance Attribute / Private / Type Ref To  IF_FPM
*      Object maintenance mode parameter
lr_parameter = me->mo_fpm->mo_app_parameter.

*      Read SYSALIAS for portal navigation
lr_parameter->get_value(
  EXPORTING iv_key   = ‘SYSALIAS’
  IMPORTING ev_value = lv_value ).
```

Important

Within a native Web Dynpro application, you must add the parameter SYSALIAS as an application parameter. Ensure that you use the same parameters on the portal or iView as well as on the Web Dynpro side. As usual, these parameters can be read from the window inbound handler (default method: HANDLEDEFAULT on window level).

5. Additional Information

5.1. Further Reading

5.1.1. Information on SAP MDG on SAP S/4HANA

- Exchange knowledge: SAP Community | Q&A | Blog
- Try SAP Master Data Governance on S/4HANA for free: Trial Version
- Try SAP Master Data Governance on S/4HANA on the SAP Cloud Appliance Library: S/4HANA 2022 FPS1
- Learn more: Latest Release | Help Portal | How-to Information | Key Presentations

5.1.2. SAP Roadmap Explorer

- Please see the roadmap for SAP Master Data Governance

5.1.3. Related Information

- Learn more: Floorplan Manager for Web Dynpro ABAP | How to Adapt FPM | FPM Blog | How-to Information | Service Mapping Tool | SAP S/4HANA Cookbook CVI

5.2. SAP Notes

In addition to the detailed explanations written in this document, please see the following SAP Notes for further important information.

Note	Description
1619534	How to Create, Enhance and Adapt FPM Applications
1637249	MDG: Information for efficient message processing
2105467	MDG Performance
2561461	Scope of support for SAP Master Data Governance (MDG)
1637249	MDG: Information for efficient message processing