

# Generating Self-Defined Functions for ALV in Web Dynpro for ABAP

## Applies to:

SAP NetWeaver 2004s – Web Dynpro for ABAP.

## Summary

This tutorial explains how to generate custom defined functions in ALV. It also explains how to associate UI elements with the defined functions and thereby embedding UI elements in ALV. This tutorial assumes that you have completed [WDA Tutorial – Programming the ALV configuration Model](#).

Author(s): Rakesh

Company: Cognizant Technology Solutions

**Created on:** 20 November 2006

## Author Bio



Rakesh is a SAP Netweaver Consultant, working with Cognizant Technology Solutions in applications based on Web Dynpro and BSP. He has knowledge in ABAP programming and Java based technologies. Currently he is working on Web Dynpro ABAP.

## Table of Contents

Applies to: .....	1
Summary.....	1
Author Bio .....	1
Step 1 – Create a New Web Dynpro Component with ALV Configuration model.....	3
Create a New Web Dynpro Component .....	3
Create Context element in Component Controller for storing the Search Criteria.....	3
Create a Context element in Component Controller for storing the Flight list .....	3
Create method for filling the Flight list.....	4
Define Component Usage.....	5
Creating View for Displaying ALV Table .....	6
Define component usage SALV_WD_TABLE in FlightView.....	6
Set data to ALV for Display .....	6
Set Functional Elements to ALV for linking functions .....	7
Step - 2 Generating Self-Defined Functions for ALV.....	7
Instantiate ALV component and get configuration model .....	7
Set the ALV header.....	8
Generate Toolbar elements .....	8
Specify properties for the Toolbar elements .....	8
Generating Objects for Self-Defined Functions and assigning Toolbar elements .....	9
Capturing Events triggered by Toolbar elements .....	9
Related Content.....	10
Disclaimer and Liability Notice.....	11

## Step 1 – Create a New Web Dynpro Component with ALV Configuration model

First a web Dynpro component with ALV usage has to be created using the ALV configuration model. A view with component usage definition has to be created.

### Create a New Web Dynpro Component

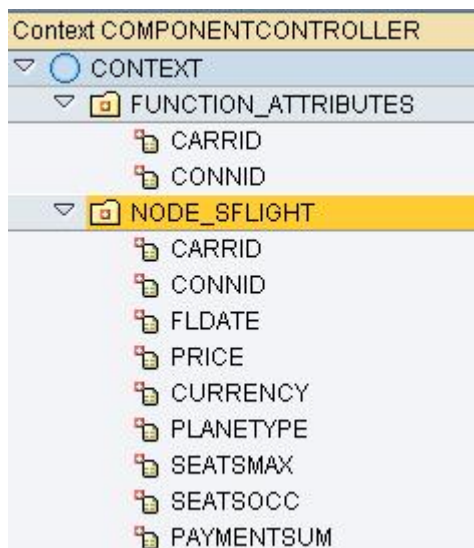
Start ABAP Workbench ( SE80 ) and create the new Web Dynpro component ZWDT\_FLIGHTLIST\_FUNCTIONS. Assign a window name.

### Create Context element in Component Controller for storing the Search Criteria

Create context node FUNCTION\_ATTRIBUTES based on the dictionary structure SFLIGHT. Click on the button *Add Attribute from Structure* and choose fields CARRID and CONNID.

### Create a Context element in Component Controller for storing the Flight list

Create context node NODE\_SFLIGHT based on the dictionary structure SFLIGHT. Set Cardinality to 0..N. Click on button *Add Attribute from Structure* and choose fields CARRID, CONNID, FLDATE, PRICE, CURRENCY, PLANETYPE, SEATSMAX, SEATSOCC and PAYMENTSUM.



Clear the DICTIONARY STRUCTURE attribute of NODE\_SFLIGHT to avoid display of all fields that are available in dictionary, in ALV.

Property	Value
<b>Nodes</b>	
Node Name	NODE_SFLIGHT
Interface Node	<input type="checkbox"/>
Input Element (Ext.)	<input type="checkbox"/>
Dictionary structure	
Cardinality	0..n
Selection	0..1
Initialization Lead Selection	<input checked="" type="checkbox"/>
Singleton	<input checked="" type="checkbox"/>
Supply Function	

### Create method for filling the Flight list

Create method FILL\_SFLIGHTS in the component controller to fill the node NODE\_SFLIGHT using the attributes CONNID, CARRID from the node FUNCTION\_ATTRIBUTES.

```

method FILL_SFLIGHTS .

data:
  node_function_attributes type ref to if_wd_context_node,
  elem_function_attributes type ref to if_wd_context_element,
  stru_function_attributes type
    if_componentcontroller=>element_Function_Attributes ,
  item_carrid like stru_function_attributes-caRRID,
  item_connid like stru_function_attributes-connID,
  ls_where(72) TYPE c,
  lt_where LIKE TABLE OF ls_where,
  lt_flights TYPE TABLE OF sflight.

data:
  node_node_sflight type ref to if_wd_context_node,
  elem_node_sflight type ref to if_wd_context_element,
  stru_node_sflight type if_componentcontroller=>element_Node_Sflight .

* navigate from <CONTEXT> to <NODE_SFLIGHT> via lead selection
  node_node_sflight = wd_context->get_child_node( name = `NODE_SFLIGHT` ).

* navigate from <CONTEXT> to <FUNCTION_ATTRIBUTES> via lead selection
  node_function_attributes = wd_context->get_child_node( name =
    `FUNCTION_ATTRIBUTES` ).

* get element via lead selection
  elem_function_attributes = node_function_attributes->get_element( ).

```

```

* get single attribute
  elem_function_attributes->get_attribute(
    exporting
      name = `CARRID`
    importing
      value = item_carrid ).

* get single attribute
  elem_function_attributes->get_attribute(
    exporting
      name = `CONNID`
    importing
      value = item_connid ).

* get element via lead selection
  elem_node_sflight = node_node_sflight->get_element( ).

* create where condition
  if not item_carrid eq ''.
    concatenate 'CARRID = '' item_carrid '' ' INTO ls_where.
    append ls_where TO lt_where.
  endif.

  if not item_connid eq '0000'.
    concatenate 'CONNID = '' item_connid '' ' INTO ls_where.
  if item_carrid ne ''.
    concatenate 'AND' ls_where into ls_where separated by space.
  endif.
  append ls_where to lt_where.
endif.

* read data from SFLIGHT
  select * from sflight into table lt_flights WHERE (lt_where).

* navigate from <CONTEXT> to <NODE_FLIGHT> via lead selection
  node_node_sflight = wd_context->get_child_node( name = `NODE_SFLIGHT` ).

* fill context node
  node_node_sflight->bind_table( lt_flights ).

endmethod.

```

## Define Component Usage

If you want to see the data within a Web Dynpro ALV, you have to define the Web Dynpro component for ALV, SALV\_WD\_TABLE as a usage component of your Web Dynpro component.

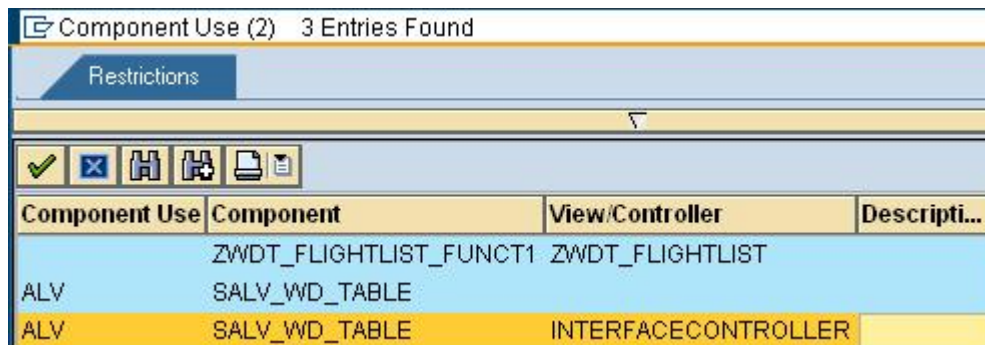
Used Web Dynpro Components	
Component Use	Component
ALV	SALV_WD_TABLE

## Creating View for Displaying ALV Table

Create View FLIGHTVIEW. In the layout of the view FLIGHTVIEW, create a *ViewContainerUIElement* called CONTAINER.

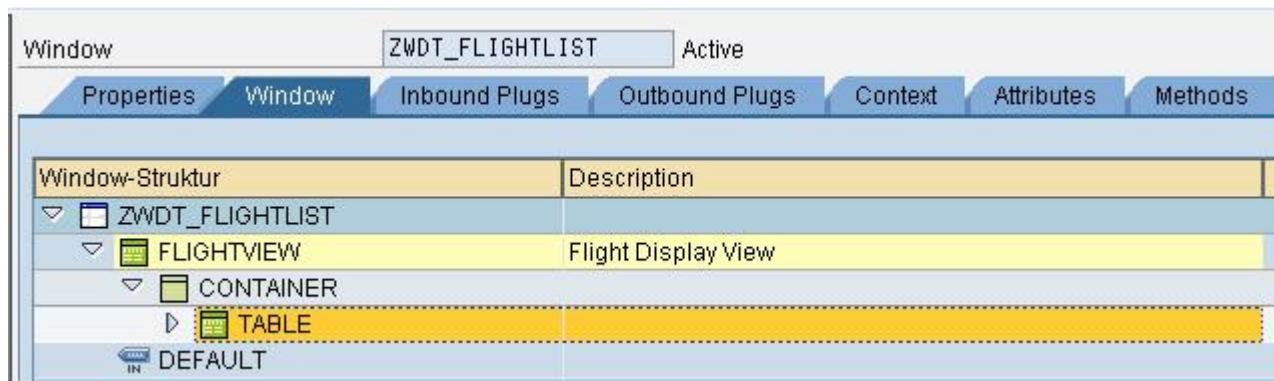
## Define component usage SALV\_WD\_TABLE in FlightView

To display ALV inside the view RESULTVIEW, it is necessary to define the component usage of SALV\_WD\_TABLE in the view. Go to the properties of view RESULTVIEW and press button *Create Controller Usage* and choose the following entry from the list on the popup:



Component Use	Component	View/Controller	Descripti...
	ZWDT_FLIGHTLIST_FUNCT1	ZWDT_FLIGHTLIST	
ALV	SALV_WD_TABLE		
ALV	SALV_WD_TABLE	INTERFACECONTROLLER	

Embed the view, FLIGHTVIEW in the window. Embed ALV table in the View container, CONTAINER.



Window-Struktur	Description
▼ ZWDT_FLIGHTLIST	
▼ FLIGHTVIEW	Flight Display View
▼ CONTAINER	
▶ TABLE	
IN DEFAULT	

## Set data to ALV for Display

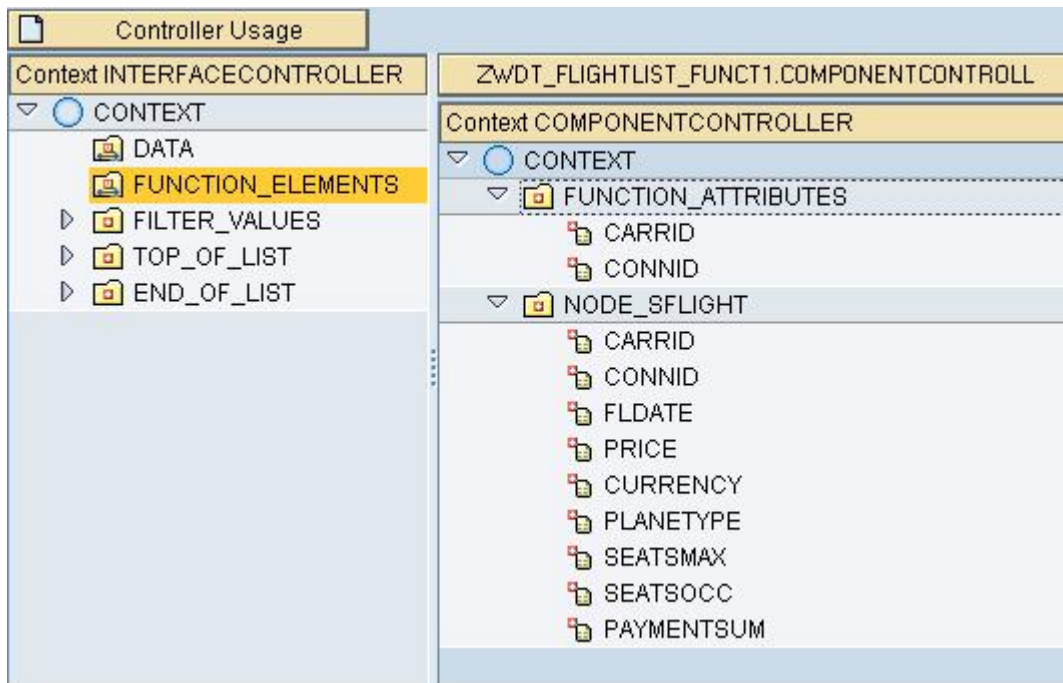
The selected flights will be inside the context node NODE\_SFLIGHT. To display them in the ALV, map the context node NODE\_SFLIGHT to the context node DATA of the ALV interface controller. Go to Web Dynpro component's node *Component Usages* -> ALV -> *INTERFACECONTROLLER\_USAGE*.

Click on the Controller Usage button. The component controller of your Web Dynpro component appears on the right side of the screen. Map the context node NODE\_SFLIGHT of your Web Dynpro component to the context node DATA of the interface controller of the ALV component.

## Set Functional Elements to ALV for linking functions

Self-defined functions like input field, that you can insert into the toolbar cause data to change when the user triggers them. For accessing the functions' data and to change them, you should map these functions to a context element of your application.

Map the context node FUNCTION\_ATTRIBUTES of your Web Dynpro component to the context node FUNCTION\_ELEMENTS of the interface controller of the ALV component.



## Step - 2 Generating Self-Defined Functions for ALV

For self-defined functions, you generate a function object with each function. Any of the possible UI Elements is specified for each function. You can generate as many function objects as you want and arrange them in the toolbar. For generating self defined functions, use WDOINIT method of the view, FLIGHTVIEW.

### Instantiate ALV component and get configuration model

Instantiate the ALV component. You can use code wizard for the purpose.

```
data: l_ref_cmp_usage type ref to if_wd_component_usage.

l_ref_cmp_usage = wd_this->wd_cpuse_alv( ).
if l_ref_cmp_usage->has_active_component( ) is initial.
  l_ref_cmp_usage->create_component( ).
endif.
```

Call the interface controller's method GET\_MODEL.

```
data: l_ref_interfacecontroller type ref to iwci_salv_wd_table .
l_ref_interfacecontroller = wd_this->wd_cpifc_alv( ).
data:
  l_value type ref to cl_salv_wd_config_table.

  l_value = l_ref_interfacecontroller->get_model(
  ).
```

### Set the ALV header

Configure the ALV's header by using GET\_HEADER method of the interface class If\_salv\_wd\_table\_settings.

```
*set header for the table
data: lr_table_settings type ref to if_salv_wd_table_settings.
data: lr_header type ref to CL_SALV_WD_HEADER.
lr_table_settings ?= l_value.

lr_header = lr_table_settings->get_header( ).
lr_header->set_text( 'FLIGHT LIST SEARCH' ).
```

### Generate Toolbar elements

Generate toolbar elements for inserting into ALV toolbar. Define the toolbar elements of the appropriate UI element type. Create Input field elements for getting CARRID and CONNID, using CREATE OBJECT. Bind the input field values with the context attributes by specifying context attributes' name in the EXPORT parameter, VALUE\_ELEMENTNAME. Create toolbar elements for button and separators ( To provide visual separation between UI elements ).

Note : The context attributes specified in the VALUE\_ELEMENTNAME parameter are mapped to FUNCTION\_ELEMENTS of the interface controller of the ALV component from the component controller.

```
DATA lr_buttonui type REF TO CL_SALV_WD_FE_BUTTON.
DATA lr_inputui1 type REF TO CL_SALV_WD_FE_INPUT_FIELD.
DATA lr_inputui2 type REF TO CL_SALV_WD_FE_INPUT_FIELD.
DATA lr_seperator1 type REF TO CL_SALV_WD_FE_SEPARATOR.
DATA lr_seperator2 type REF TO CL_SALV_WD_FE_SEPARATOR.

CREATE OBJECT lr_inputui1 EXPORTING VALUE_ELEMENTNAME = 'CARRID'.
CREATE OBJECT lr_inputui2 EXPORTING VALUE_ELEMENTNAME = 'CONNID'.

CREATE OBJECT lr_buttonui.
CREATE OBJECT lr_seperator1.
CREATE OBJECT lr_seperator2.
```

### Specify properties for the Toolbar elements

Set text for the button toolbar element. Set labels for the input field elements.



```

lr_buttonui->SET_TEXT( 'Search Flights' ).
lr_inputui1->SET_LABEL_TEXT( ' AIRLINE :' ).
lr_inputui2->SET_LABEL_TEXT( ' FLIGHT NO :' ).

```

### Generating Objects for Self-Defined Functions and assigning Toolbar elements for the objects

Generate a function object of class CL\_SALV\_WD\_FUNCTION for each function you create. When you generate a self-defined function, specify a unique ID, for addressing the function.

Function objects are created using CREATE\_FUNCTION method of the interface class IF\_SALV\_WD\_FUNCTION\_SETTINGS. Specify the toolbar element for the functions by using the method SET\_EDITOR.

```

DATA input1 TYPE REF TO CL_SALV_WD_FUNCTION.
input1 = l_VALUE->IF_SALV_WD_FUNCTION_SETTINGS~create_function( id =
'LINPUT1' ).
input1->SET_EDITOR( lr_inputui1 ).

DATA seperator1 TYPE REF TO CL_SALV_WD_FUNCTION.
seperator1 = l_VALUE->IF_SALV_WD_FUNCTION_SETTINGS~create_function( id =
'LS' ).
seperator1->SET_EDITOR( lr_seperator1 ).

DATA input2 TYPE REF TO CL_SALV_WD_FUNCTION.
input2 = l_VALUE->IF_SALV_WD_FUNCTION_SETTINGS~create_function( id =
'LINPUT2' ).
input2->SET_EDITOR( lr_inputui2 ).

DATA seperator2 TYPE REF TO CL_SALV_WD_FUNCTION.
seperator2 = l_VALUE->IF_SALV_WD_FUNCTION_SETTINGS~create_function( id =
'LS2' ).
seperator2->SET_EDITOR( lr_seperator2 ).

DATA button1 TYPE REF TO CL_SALV_WD_FUNCTION.
button1 = l_VALUE->IF_SALV_WD_FUNCTION_SETTINGS~create_function( id =
'LBUTTON' ).
button1->SET_EDITOR( lr_buttonui ).

```

### Capturing Events triggered by Toolbar elements

Create a event handler method ON\_SEARCH in the view, RESULTVIEW. Specify event, ON\_FUNCTION of the interface controller of ALV for the event handler method. The importing parameter r\_param of the method contains unique ID of the function for which the toolbar element is assigned.

```

method ON_SEARCH .

    DATA: temp TYPE string.
    temp = r_param->id.
    IF temp = 'LBUTTON'.
        wd_comp_controller->fill_sflights( ).
    ENDIF.

endmethod.

```

Create an application and save the component. Activate and test the component. The search criteria and the display for the Flight application will be present inside the ALV itself.

**FLIGHT LIST SEARCH**

Excel Print Version AIRLINE : LH FLIGHT NO : 0000 Search Flights Filter Settings

Date	Price	Currency	Pl.type	Capacity	Occupied	Total
02.04.2005	666,00	EUR	A310-300	280	271	210.875,58
30.04.2005	666,00	EUR	A310-300	280	263	202.490,64
28.05.2005	666,00	EUR	A310-300	280	266	208.684,44
25.06.2005	666,00	EUR	A310-300	280	267	209.723,40
23.07.2005	666,00	EUR	A310-300	280	271	210.282,84
20.08.2005	666,00	EUR	A310-300	280	261	205.081,38
28.08.2005	666,00	EUR	A310-300	280	271	211.568,22
17.09.2005	666,00	EUR	A310-300	280	267	209.203,92
28.09.2005	666,00	EUR	A310-300	280	265	207.772,02
15.10.2005	666,00	EUR	A310-300	280	272	209.790,00

Row 1 of 85

## Related Content

Please include at least three references to SDN documents or web pages.

[WDA Tutorial - Simple Example for Using ALV](#)

[WDA Tutorial – Programming the ALV configuration Model](#)

[WDA Official Documentation](#)

## Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.