

**How-to Guide
SAP NetWeaver '04**



How To... Run Planning Sequences on Save and other events (WEB)

Version 1.00 – September 2004

**Applicable Releases:
SAP NetWeaver '04
(BW 3.5 SPS 5)**

© Copyright 2004 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C[®], World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data

contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver "How-to" Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

1 Scenario

When creating a user interface for planning there is often a need to execute planning sequences automatically on certain events. For example a planning sequence should be executed when pressing the save button in order perform a currency conversion.

When transferring SAPGui based planning folders to a Web Interface Application you find that the execution of functions *before layout display* and *on save* are not supported. By following through this document these features are added and planning folders can be transferred more easily.

Please keep in mind that such kinds of automatic functions will typically increase the response time. In particular the layout related execution occurs quite frequently (on each submit of data or server roundtrip).

2 Introduction

This document describes how to enhance the Web Interfaces to automatically trigger a global planning sequence ...

- ... by pressing the save button.
- ... before a layout is executed.

To enable this feature a mapping of planning sequences to layouts and save buttons is required. One option given is to reuse the standard function button for carrying the sequence names. Following this approach you get things like F4-help etc. for free and you stay within the standard customizing.

The solution uses the Web Interface Extension (the so called 'exit class') to enhance the input and output processing of the Web Interface.

Once a planning sequence has been executed you will see the status in the message tray. In case of errors the complete list of single functions is shown and the following step is not executed.

Assignments using function buttons

The assignment of a planning sequence is done by adding a function button to the Web Interface. The attribute contains the name of the global planning sequence whereas the component name follows the rule indicated below:

Component Name	Suffix	Name of function button
MyLayout	_before	MyLayout_before
SaveButton	_save	SaveButton_save

3 The Step By Step Solution

The step-by-step solution comes with two sections: first we have to enhance the standard input and output processing of the Web Interface. This is done on the backend and described in section 3.1. Additionally we have to adjust the customizing of the Web Interfaces in order to enable the new functionality for a specific Web Interface. This is described in section 3.2 .

3.1 Enhance input and output processing

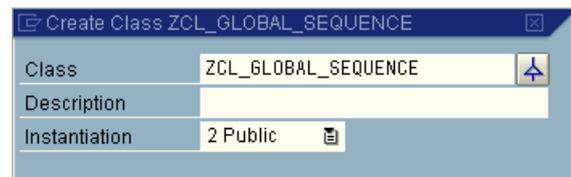
Some technical background information:

- When creating a Web Interface using the Web Interface Builder (transaction bps_wb) the system generates a BSP page based on the customizing given. The BSP event handlers are implemented in the class `cl_upwb_bsp_appl`.
- By creating subclasses and using the redefinition capability of ABAP OO, the standard behavior can be changed and new functionalities added.
- During the input processing the system creates a list of tasks to be done (which components to use and which actions to perform) and dispatches the work to the different component. In order to add the validation feature, we redefine the dispatcher method.

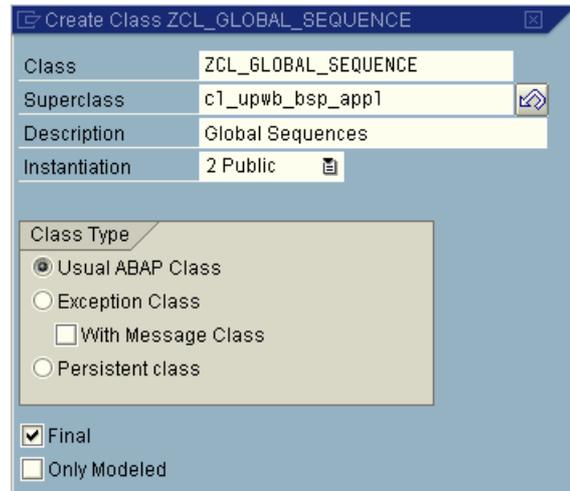
1. Call transaction SE24, enter the new class name and press Create.



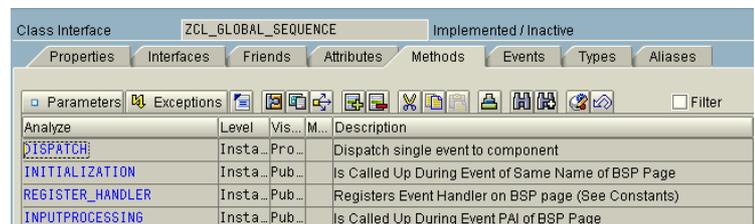
2. Press the  Button on the pop-up to get the additional input field for the super class.



- Complete the pop-up as shown in the screenshot and press Save.



- Now you should see this screen. Select the method Dispatch and press the button (when using SAP Gui 6.40) to redefine the method.



- Now an editor opens showing the following default ABAP code. Delete the coding and insert the ABAP coding given in section 4.1 Source Code: Method dispatch of this How To.

```
method DISPATCH.
*CALL METHOD SUPER->DISPATCH
* EXPORTING
*   IS_HANDLER =
* RECEIVING
*   R_SUBRC =
*
*
endmethod.
```

- Note:** When executing a planning sequence the status is shown in error and success case. By uncommenting ... the status is shown in error case only.

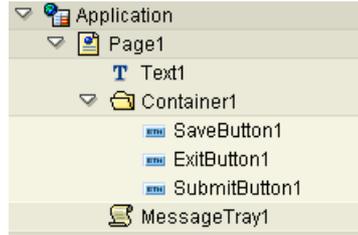
```
* If you like to see message in error cases only, uncomment the
if ... endif
* IF 1_subrc <> 0.
  tr_component->mo_msg_log->add( EXPORTING it_bapiret =
1t_msg_log ).
* ENDIF.
```

- After activating the class (press button) we have now created a subclass of the standard implementation and we have done the back-end part of the enhancement. Congratulations!

3.2 Assignments using function buttons

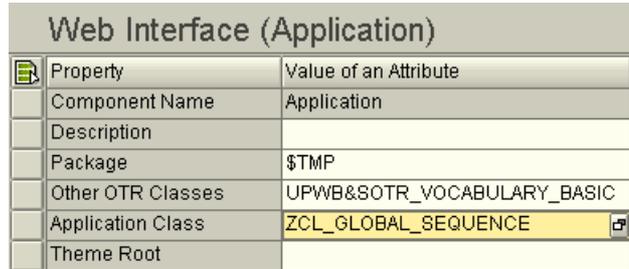
- Start the Web Interface Builder using transaction BPS_WB.

- Open the Web Interface to be enhanced.
The application used for this example has the name *Application*.

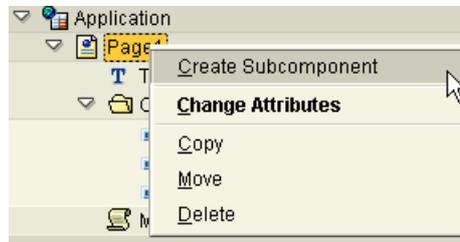


- Double click on the root node to open the property section for the *Application*.

Replace the application class `cl_upwb_bsp_appl` by `zcl_global_sequence`, the name of the subclass created before.



- Right click on the page node to add a new subcomponent.



- Chose the text component.

Exit Pushbutton	BUT_EXIT
Function Pushbutton	BUT_FUNCTION
Group	GROUP

- Concatenate the component name of the save button with the suffix `_save` (or the name of the layout component with the suffix `_before`).



- Set the property `Visible` to `false` and choose a global planning sequence.

Parameter Group	
Global Planning Sequence	mySequence
Screen (Path)	
Quick Info Text	
Visible	false

- (Re-) Generate the planning application by pressing  button.

4 Appendix

4.1 Source Code: Method dispatch

METHOD dispatch.

```
CONSTANTS:
  c_layout_class  TYPE string VALUE 'LAYOUT',
  c_layout_method TYPE string VALUE 'API_GET_DATA',
  c_save_class    TYPE string VALUE 'BUT_SAVE',
  c_save_method   TYPE string VALUE 'SUBMIT'.

DATA:
  lr_component TYPE REF TO if_upwb_c_component,
  lr_convert   TYPE REF TO if_upwb_c_component,
  lr_lay       TYPE REF TO if_upwb_c_layout2,
  lr_func      TYPE REF TO if_upwb_c_but_function,

  l_class      TYPE string,
  l_id         TYPE string,
  l_id_seq     TYPE string,

  l_bundle TYPE upf_y_bundle,
  l_subrc   LIKE sy-subrc,
  lt_msg_log TYPE TABLE OF bapiret2.

lr_component ?= is_handler-component.
l_class      = lr_component->get_class( ).
l_id        = lr_component->get_id( ).

* Do we have the right component and method for starting sequence?
IF l_class = c_layout_class AND is_handler-method = c_layout_method.
  lr_lay ?= lr_component.
  IF lr_lay->is_ready_for( cl_upwb_bsp_appl=>ce_get_data ) = cb_true.
    CONCATENATE l_id 'before' INTO l_id_seq.
  ENDIF.
ENDIF.

* Do we have the right component and method for starting sequence?
IF l_class = c_save_class AND is_handler-method = c_save_method.
  CONCATENATE l_id '_save' INTO l_id_seq.
ENDIF.

IF NOT l_id_seq IS INITIAL.

* Get name of sequence from function component
lr_convert = cl_upwb=>get_component( l_id_seq ).
IF lr_convert IS BOUND.
  lr_func ?= lr_convert.
  l_bundle = lr_func->get_gl_sequence( ).
ENDIF.

* If no sequence is found, l_bundle is initial
IF NOT l_bundle IS INITIAL.
  CALL FUNCTION 'API_SEMBPS_GLSEQUENCE_EXECUTE'
    EXPORTING
      i_sequence = l_bundle
    IMPORTING
      e_subrc    = l_subrc
  TABLES
    etk_return  = lt_msg_log.

* If you like to see message in error cases only, uncomment the if ... endif
* IF l_subrc <> 0.
  lr_component->mo_msg_log->add( EXPORTING it_bapiret = lt_msg_log ).
* ENDIF.

* Valid from BW3.5 SP5 onwards
cl_upwb=>raise_buffer_changed( 'buffer' ).
* Valid until BW3.5 SP4 and for SEM 3.1, 3.2, 3.5.
* cl_upwb=>set_parameter( param = 'buffer_changed' value = 'X' ).
IF l_subrc <> 0. r_subrc = 10. ENDIF.
ENDIF.

IF r_subrc IS INITIAL.
  CALL METHOD super->dispatch
    EXPORTING
      is_handler = is_handler
    RECEIVING
      r_subrc    = r_subrc.
ENDIF.

ENDMETHOD.
```

www.sdn.sap.com/irj/sdn/howtoguides