How-To Guide

How To Add Application Content to the Warehouse Management Monitor

© Copyright 2022 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, Excel, Outlook, PowerPoint, Silverlight, and Visual Studio are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, z10, z/VM, z/OS, OS/390, zEnterprise, PowerVM, Power Architecture, Power Systems, POWER7, POWER6+, POWER6, POWER, PowerHA, pureScale, PowerPC, BladeCenter, System Storage, Storwize, XIV, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, AIX, Intelligent Miner, WebSphere, Tivoli, Informix, and Smarter Planet are trademarks or registered trademarks of IBM Corporation.

Linux is the registered trademark of Linus Torvalds in the United States and other countries.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are trademarks or registered trademarks of Adobe Systems Incorporated in the United States and other countries.

Oracle and Java are registered trademarks of Oracle and its affiliates.

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.

Apple, App Store, iBooks, iPad, iPhone, iPhoto, iPod, iTunes, MultiTouch, Objective-C, Retina, Safari, Siri, and Xcode are trademarks or registered trademarks of Apple Inc.

IOS is a registered trademark of Cisco Systems Inc.

RIM, BlackBerry, BBM, BlackBerry Curve, BlackBerry Bold, BlackBerry Pearl, BlackBerry Torch, BlackBerry Storm, BlackBerry Storm2, BlackBerry PlayBook, and BlackBerry App World are trademarks or registered trademarks of Research in Motion Limited.

Google App Engine, Google Apps, Google Checkout, Google Data API, Google Maps, Google Mobile Ads, Google Mobile Updater, Google Mobile, Google Store, Google Sync, Google Updater, Google Voice, Google Mail, Gmail, YouTube, Dalvik and Android are trademarks or registered trademarks of Google Inc.

INTERMEC is a registered trademark of Intermec Technologies Corporation.

Wi-Fi is a registered trademark of Wi-Fi Alliance.

Bluetooth is a registered trademark of Bluetooth SIG Inc.

Motorola is a registered trademark of Motorola Trademark Holdings LLC.

Computop is a registered trademark of Computop Wirtschaftsinformatik GmbH.

SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, StreamWork, SAP HANA, 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 other countries.

Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. Business Objects is an SAP company.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase Inc. Sybase is an SAP company.

Crossgate, m@gic EDDY, B2B 360°, and B2B 360° Services are registered trademarks of Crossgate AG in Germany and other countries. Crossgate is an SAP company.

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.

Disclaimer:

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.

Document History

Document Version	Description
2.0	New chapter entitled *Form View for New Nodes* (SAP S/4HANA only and for release 2022 or later)
1.0	First official release of this guide

Typographic Conventions

Type Style	Description
Example Text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
`Example text`	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

Icons

Icon	Description
	Caution
	Important
	Note
	Recommendation or Tip
	Example

Contents

Document History ii
1. Introduction 6
1.1 The Proof of the Pudding is in the Eating - Intention of How-To Guide 6
2. Prerequisites 6
3. Overview 7
3.1 Providing Content for Warehouse Management Monitors 8
4. Enhancement of Existing Monitor Node Content 8
4.1 Prerequisite Steps 8
4.1.1 Create Own Monitor by Copying SAP Template 9
4.1.2 Create Function Group for Customer Function Modules 11
4.2 Example: Display Product Description for Outbound Delivery Order Items 12
4.2.1 Solution Proposal 12
4.2.2 Required Steps 12
4.2.3 Identification of Node Data 13
4.2.4 Copy Node to Customer Node 15
4.2.5 Copy Node Profile 16
4.2.6 Copy and Enhance DDIC Structures 17
4.2.7 Create Wrapper Function Module 18
4.2.8 Copy and Adjust Screen (Form Layout) 21
4.2.9 Change Customizing Settings for Monitor Node 22
4.2.10 Testing 24
4.3 Advanced Example: Extend Select Options for Deliveries and Display Results 26
4.3.1 Solution Proposal 26
4.3.2 Required Steps 26
4.3.3 Identification of Node Data 27
4.3.4 Copy Node to Customer Node 27
4.3.5 Copy Node Profile 27
4.3.6 Copy and Enhance DDIC Structures 28
4.3.7 Create Function Module 28
4.3.8 Copy and Adjust Screen (Form Layout) 30
4.3.9 Change Customizing Settings for Monitor Node 31
4.3.10 Testing 32
5. Add New Methods to Monitor 34
5.1 Example: New Method for Unassigning Outbound Delivery Order Items from Wave 34
5.1.1 Solution Proposal 34
5.1.2 Required Steps 34
5.1.3 Prerequisites for Function Modules used in Object Class Methods 35
5.1.4 Sample Function Module ZHTG_WAVEMON_UNASSIGN 35
5.1.5 Define New Custom Object Class ZHTGOI 38

5.1.6 Maintain Object Class Methods for Custom Object Class ZHTGOI 39
5.1.7 Define Method Presentation of Custom Object Class Method 41
5.1.8 Exchange Object Class of Monitor Node against Custom Object Class 42
5.1.9 Testing 42
6. Add New Application Content 43
6.1 Customize Own Monitors 43
6.2 Example: Create Custom Monitor for New Custom Nodes 43
6.2.1 Required Steps 44
6.2.2 Define Custom Monitor ZHTG_S 44
6.2.3 Create New Category ZHTG_C 45
6.2.4 Define New Object Class ZHTGID 46
6.2.5 Define New Object Class Methods 47
6.2.6 Define Methods Presentation 47
6.2.7 Define Node Profiles for Monitor Tree Nodes 49
6.2.8 Define Monitor Tree Nodes 50
6.2.9 Define Node Hierarchy for Monitor ZHTG_S 51
6.2.10 Testing New Custom Monitor ZHTG_S 53
6.3 Function Modules for New Application Content 54
6.3.1 Selection Screens 54
6.3.2 Interface 55
6.3.3 Mandatory Processing Steps 56
6.4 Form View for New Nodes (SAP S/4HANA 2022 or Later) 63
7. Maintain Hotspots for Navigation 64
7.1 Example: Hotspot for Displaying Outbound Delivery Order Items in Enhanced Monitor Node 65
7.1.1 Define Hotspot 66
7.1.2 Map Service Keys Fields 66
7.2 Example: Maintain Hotspot Navigation to Inbound Delivery 68
7.2.1 Define Hotspot 68
7.2.2 Map Service Keys Fields 68
8. Appendix 69
8.1 Sample Coding 69
8.1.1 Master Program SAPLZHTG (Function Pool) 69
8.1.2 Include LZHTGTOP 69
8.1.3 Include LZHTGP01 71
8.1.4 Include LZHTGP02 73
8.1.5 Function Module ZHTG_WHRHEAD_MON_XT 108
8.1.6 Text Symbols and Selection Texts of Function Group ZHTG 124
8.1.7 Function Module ZHTG_WAVEMON_UNASSIGN _XT– Extended Check for Typing of IT_DATA 126

1. Introduction

Note: The information contained in this how-to guide applies to both SAP EWM and EWM in SAP S/4HANA.

This document provides you with examples how the content of the EWM warehouse management monitor can be enhanced and adjusted to fulfill your business needs. Enhancement possibilities are explained with custom coding examples that are based on delivered SAP standard content. Each example focuses on specific enhancement features.

Learn how to use these features by implementing corresponding custom coding elements that are described in detail in this document. Get familiar with the Customizing settings for the warehouse management monitor that allow you to build your own warehouse management monitor content. This how-to guide allows a step-by-step approach, starting with examples that are easy to understand and easy to implement.

1.1 The Proof of the Pudding is in the Eating - Intention of How-To Guide

SAP provides a monitor tree that consists of predefined nodes, categories, node profiles, and predefined object classes. The SAP standard monitor can be used as a basis for customer-specific monitor (trees) that are tailored to fulfill customers" needs.

The proof of the pudding is in the eating: Follow the step-by step descriptions given in this guide to get familiar with concepts for enhancing monitor content. Working through these examples allows you to gain knowledge that you require for creating your own custom content.

There are various ways to extend content of your warehouse management monitors. The following table provides you with an overview about enhancement possibilities that are covered in this document:

Enhancement Option	Chapter
Copy SAP standard monitor node and enhance content of copied node	**4**
Maintain hotspots for easier navigation	**7**
Add additional methods to monitor nodes	**5**
Add new application content by creating and adding new monitor nodes	**6**

2. Prerequisites

Technical prerequisites:

- SAP EWM system (release SCM 7.02 and higher) containing sample data for inbound and outbound processes
    - EWM system should be connected to SAP ERP system
- Authorizations
    - for creating coding elements in your customer namespace, that is: function modules, screens
    - for maintaining own custom warehouse management monitor elements

Development expertise:

- ABAP development skills and experience with object-oriented ABAP (ABAP O-O)

Business context:

- Know-how about basic concepts and business scenarios of SAP EWM

- Usage of EWM warehouse management monitor

Starting documentation: SAP Library chapter "Warehouse Management Monitor"

3. Overview

In general, the warehouse management monitor is the central tool for displaying EWM documents and processes.

Primarily, there are two views – one for the document view and one for the process view. The UI consists of a hierarchical navigation tree and two sub screens. One sub screen is used to display data of the parent objects. The other sub screen displays the data of the child objects. Within both sub screens, it is possible to toggle between a list and form view.

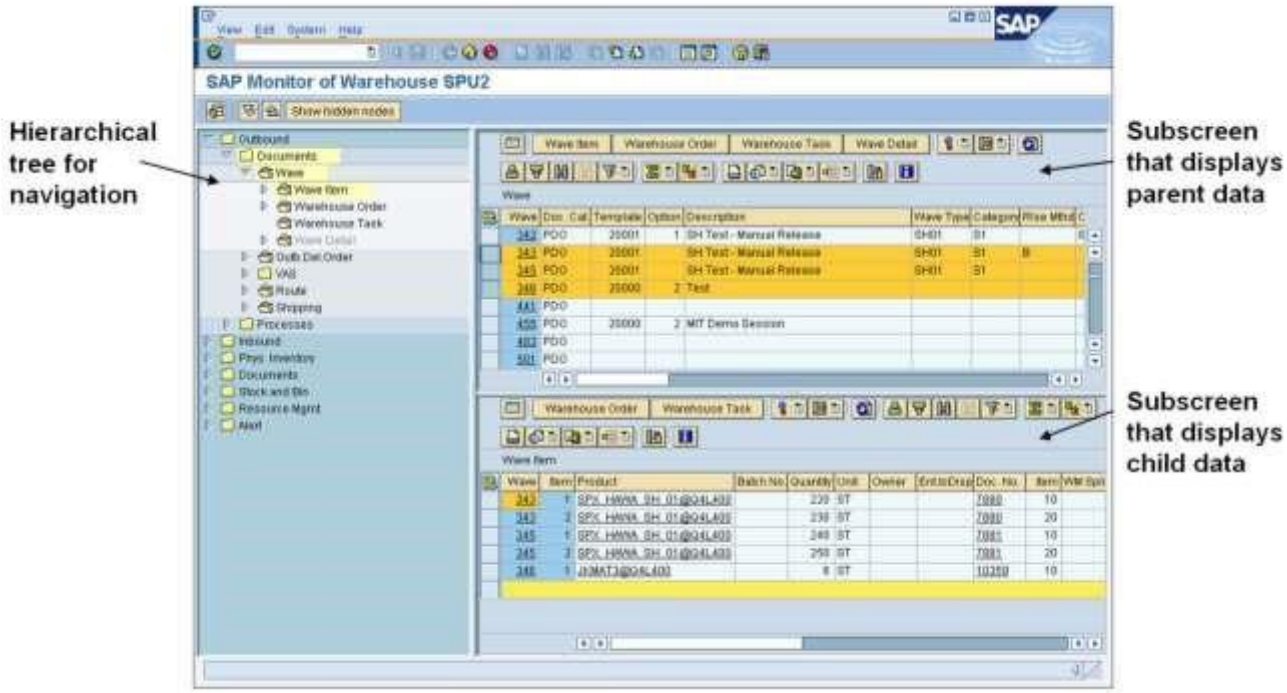


Figure: Displaying Wave Item Data in Result List of Warehouse Management Monitor

The navigation within the monitor can be performed via the hierarchical tree on the left and by using a drill down functionality via the push buttons for the children in the ALV application toolbar.

Methods can be specified that trigger a certain function for each object class. For example, you could consider adding a method to confirm warehouse tasks (WTs) in the background for the object class TO. These methods will be accessible via additional pushbuttons in the ALV toolbar.

Additionally, it is possible to define hotspots for output fields in the ALV list. Using these hotspots, the corresponding UI can be called for an object ID (for example, clicking on WT number calls the WT display transaction).

The monitor framework offers the means by which the monitor can easily be modified according to the customers" needs. It is possible either to create a completely new monitor or modify the SAP standard monitor in Customizing. These changes will be valid for all users. In addition, it is possible to create user-specific monitor trees. In the productive system, every user can hide nodes or complete branches that they do not want to use. A user can create variant nodes for queries they access very often. These comprise a specific set of selection criteria. Additionally, the user can create nodes with a given layout for the ALV list view. Of course, a combination of both is also possible.

3.1 Providing Content for Warehouse Management Monitors

Warehouse management monitors contains various nodes that are arranged hierarchically. Setting up hierarchical tree structures for nodes is done with the help of Customizing settings. The main elements of this structure are:

- Monitor tree
  - Visualization of hierarchical relations of tree nodes
- Categories
  - Tree nodes that are used for grouping tree nodes. Categories do not have node profiles.
- Nodes
  - Nodes are elements ("leaves") of the monitor tree hierarchy. Either they carry functions to be executed on EWM objects or they are used for the grouping of tree content only (category nodes). Nodes with executable functions are assigned to node profiles.
  - Nodes use special function modules for selecting data from EWM content. The results of data selections are displayed in the ALV list.
- Node profiles
  - Collection of information that defines way how data selection is executed and how results of selection are displayed. Follow-on functions of selected result list entries are processed with the help of object class methods.
- Object classes
  - Bundle operations that are invoked from the warehouse management monitor, in separate object class methods. These methods can be performed on one or more selected objects from the result list.

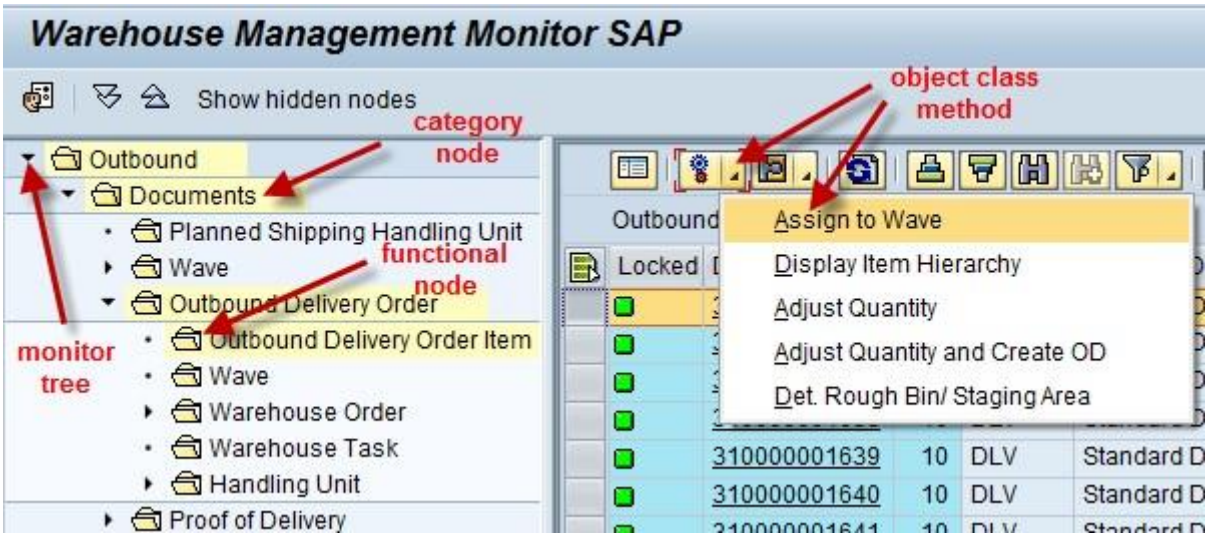


Figure: SAP Standard Monitor Tree showing Category Nodes, Functional Nodes, and Object Class Methods

Recommendation

More information about Customizing settings for warehouse management monitors can be found in the Customizing documentation under *Extended Warehouse Management -> Monitoring -> Warehouse Management Monitor*.

4. Enhancement of Existing Monitor Node Content

4.1 Prerequisite Steps

As a basis for implementation, the SAP monitor template is copied to a customer-specific monitor.

4.1.1 Create Own Monitor by Copying SAP Template

Warehouse management monitors are defined in SAP Customizing. Open the Customizing node *Extended Warehouse Management → Monitoring → Warehouse Management Monitor*.

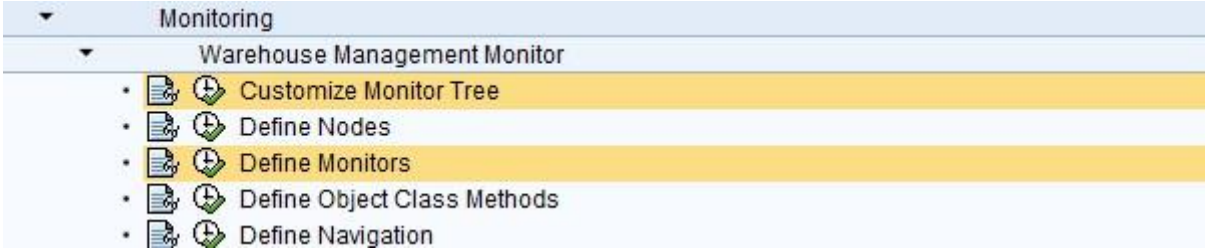


Figure: Customizing Activities for Creation of Warehouse Management Monitors

SAP offers two possibilities for customer-specific monitors:

- Customizing activity *Customize Monitor Tree* - graphical tool, supporting drag and drop
- Customizing activity *Define Monitors* - tabular view maintenance

The steps to copy the SAP monitor content to own customer monitor are executed with the help of the Customizing activity *Customize Monitor Tree*. A screen appears that contains two areas that show monitor tree hierarchies. By default, the SAP standard monitor tree hierarchy is displayed in left area of screen. If this is not the case, then load the standard monitor tree by pressing icon *Display Monitor*.

Figure: Load SAP Standard Monitor Node Hierarchy

Now create a new monitor by dragging folders from the SAP monitor on the left and dropping them on the new monitor icon on the right side. Save new monitor: Provide values for the fields *Warehouse*, *Monitor*, and *Text*:

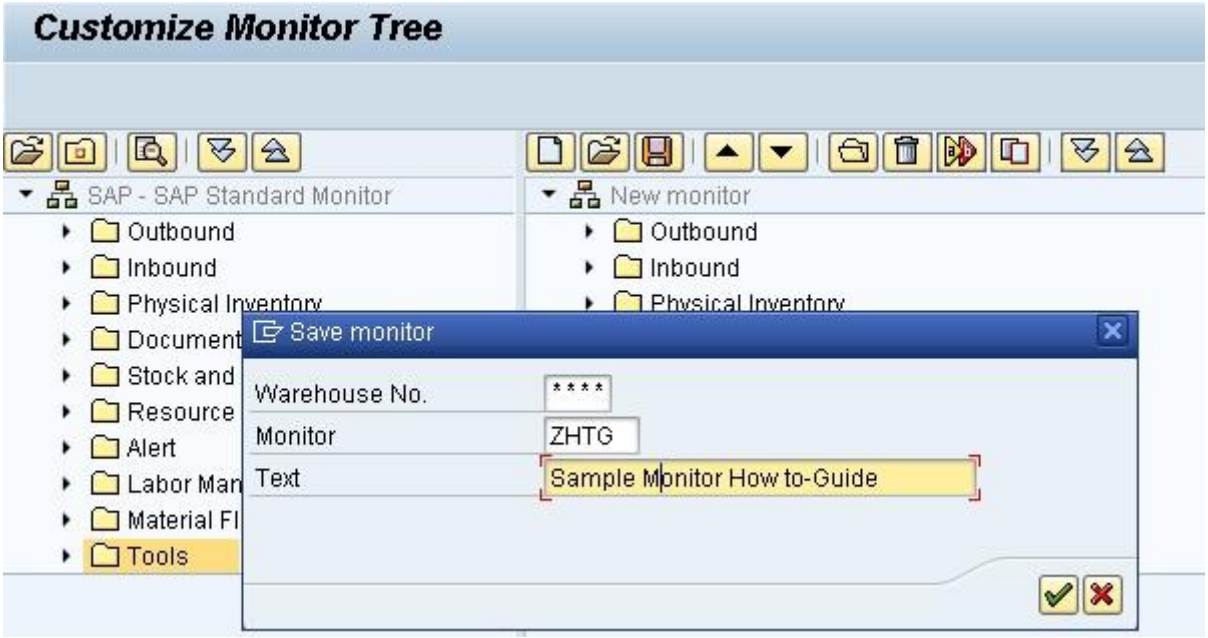


Figure: Copy SAP Standard Monitor to Custom Monitor ZHTG

Note

The copying function does not create copies of each node that is represented in the tree structure. As a consequence, the copied monitor consists of the same nodes as the SAP standard monitor. Changes to the SAP standard nodes are not recommended and will impact all monitors that use a changed standard node. Therefore, node enhancement should be carried out on the copied monitor nodes (as described in the following chapters).

Nodes that are assigned to parent nodes as child nodes require the possibility to transfer data of the selected parent nodes while selecting child nodes. Therefore the following requirements have to be fulfilled:

- Parameter interface of function module (child node) contains importing parameter IT_PARENT_DATA for storing data of selected parent nodes
- Parameter interface of function module (parent node) contains exporting parameter ET_DATA that uses same table type as parameter IT_PARENT_DATA (child node)

If these requirements are not fulfilled, the system stops assigning the child node and raises an error message.

Note

The checking logic is implemented in subroutine PLAUS_CHECK of include /SCWM/RMONMAINT_FORM1 (main program: /SCWM/R_MONITOR_MAINT)

4.1.1.1 Copy Existing Object Class Methods of SAP Tree

For completeness copy those object class methods that belong to copy template: Execute the Customizing activity *Define Object Methods*:

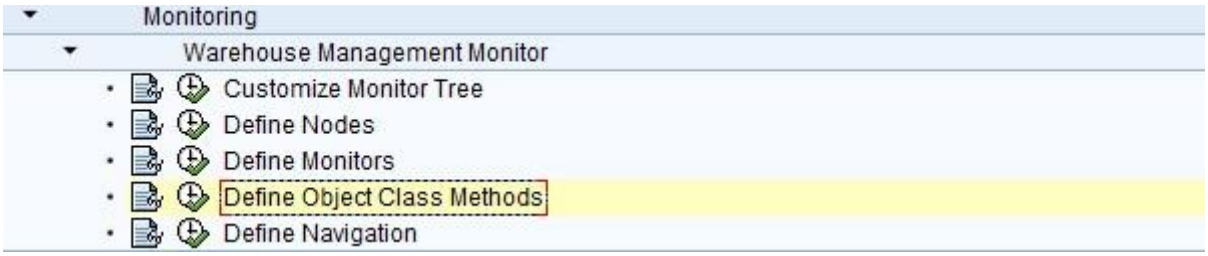


Figure: Customizing Activity for Defining Object Class Methods for Monitor Nodes

On the following screen double-click on the folder *Define Methods Presentation*. Mark the relevant entries and press the icon *Copy As* (F6):

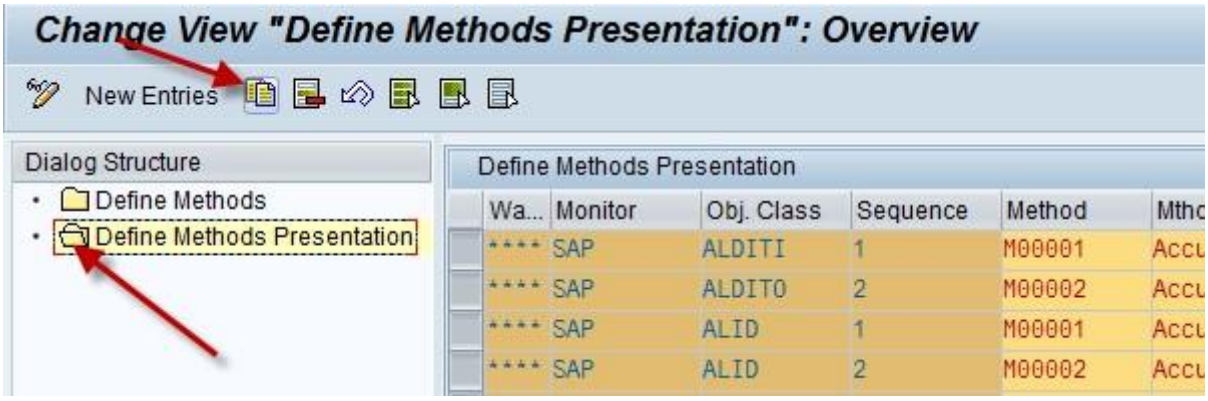


Figure: Copy Object Class Method Presentation

Change the values in columns *Warehouse* and *Monitor* accordingly:

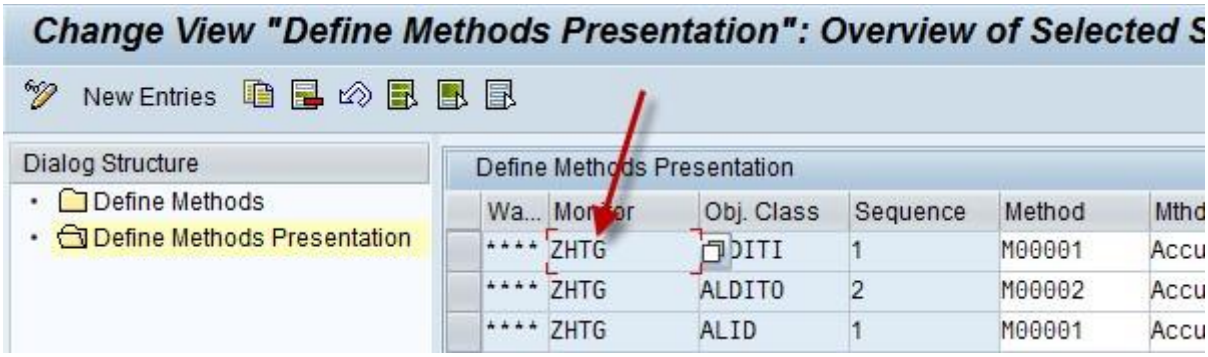


Figure: Copied Object Class Methods for Custom Monitor ZHTG Save

your changes.

4.1.2 Create Function Group for Customer Function Modules

Start the Object Navigator (transaction SE80) and create a new function group *ZHTG* for enhanced customer function modules:

- Short text: *Function Group for How To-Guide Samples*

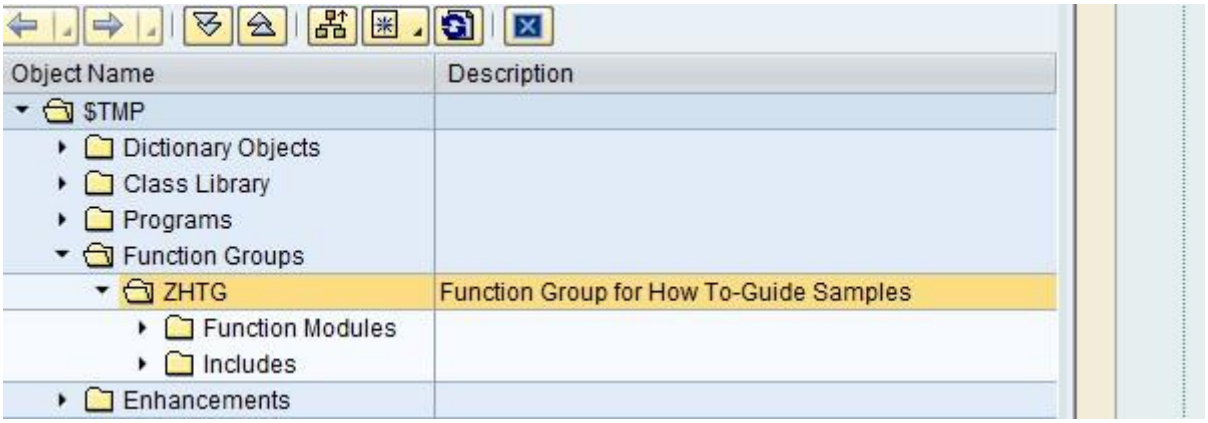


Figure: Function Group for Extended Function Modules of Monitor Node Content

Save and activate your changes.

4.2 Example: Display Product Description for Outbound Delivery Order Items

Short descriptions of EWM products are often used to provide detailed information about products to warehouse workers. Unfortunately the product short description of outbound delivery order items is not displayed when the corresponding monitor node is double-clicked:

Figure: Missing Product Description for Outbound Delivery Items (List View)

Neither the list view nor the form view display the required product short description:

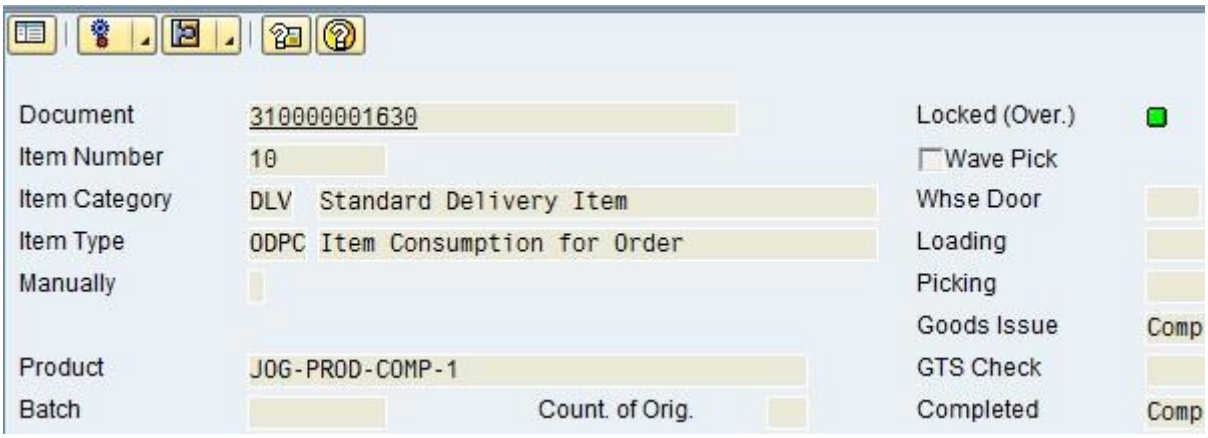


Figure: Missing Product Description for Outbound Delivery Items (Form View)

4.2.1 Solution Proposal

The customer solution shall fulfill the following requirements:

- Display the product short description for each outbound delivery order item in list view and form view
- Display the product name and short description directly after the column that shows the description of the item category
- Display the product short description when the user double-clicks on the folder *Output Delivery Order Item*
- Display the product short description when the user navigates from the selected outbound delivery orders to the detailed view of outbound delivery order items

4.2.2 Required Steps

The following steps are required to display the product short description in the above-mentioned monitor node:

- Identify the node ID of the node *Outbound Delivery Order Item* and copy the node
- Identify the associated node profile and copy the profile

- Identify the DDIC structures used for displaying the item data and identify the function module that collects the item data
- Copy the DDIC structures into the customer namespace.
    - Append the field that stores the product short description
- Create a wrapper function module in the customer namespace that calls the SAP standard function module for selecting outbound delivery order item data.
    - Add coding that allows reading of short descriptions for all selected products.

      Recommendation

      Creating a wrapper function module keeps implementation efforts low compared with copying and enhancing the SAP standard function module for selecting outbound delivery order item data. Creation of own function modules for monitor node usage is described in detail in chapter 6.1.
- Copy the screen that displays the data in form view into the customer function group.
    - Add a label and output field for the product short description.
- Change the Customizing settings for the monitor node:
    - Exchange function module, DDIC structures, screen program, and screen number in the copied node profile
    - Update the settings for the selected monitor node
- Test changes

  Call the warehouse management monitor using the monitor hierarchy that contains the enhanced monitor node

4.2.3 Identification of Node Data

Get information about the node ID of the selected monitor node: Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Customize Monitor Tree*:

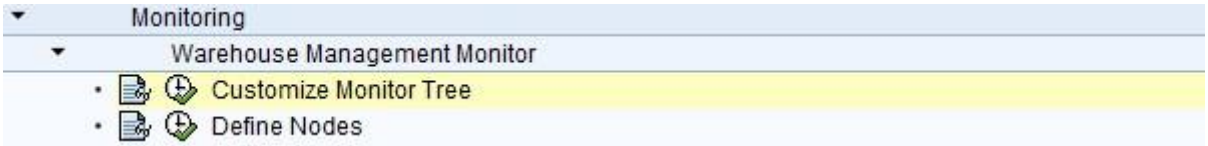


Figure: Customizing Activity for Customizing Warehouse Management Monitor Tree

On the following screen press icon *Display monitor* and choose your monitor. The node hierarchy is loaded:

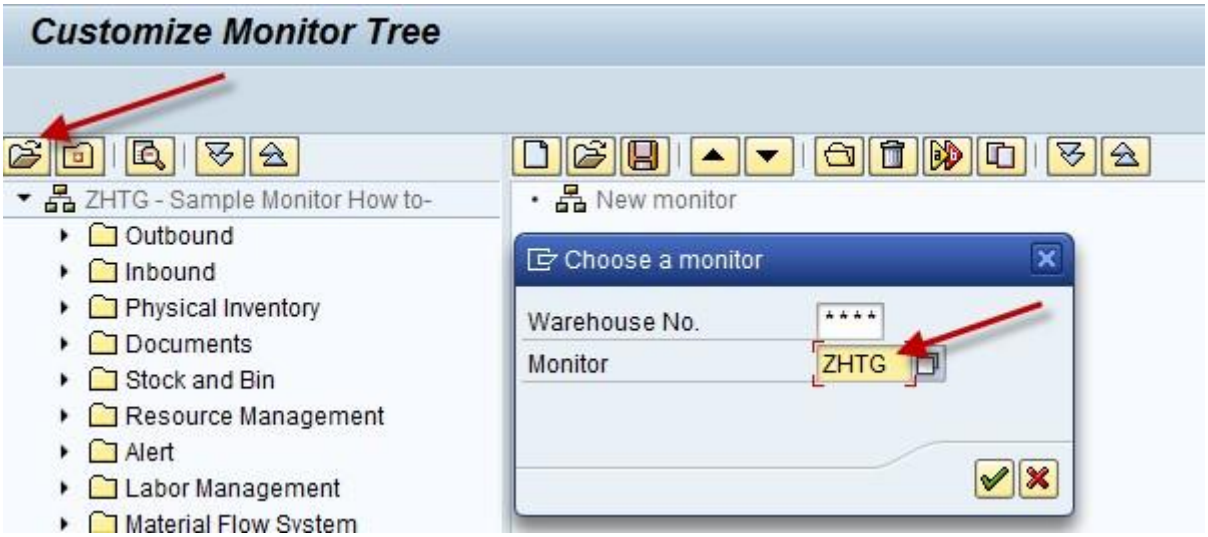


Figure: Load Copied Warehouse Management Monitor

4.2.3.1 Display Node Data of Selected Node

Expand the hierarchy below the folder *Outbound* and double-click on the node *Outbound Delivery Order Item*. The node data is shown in a popup screen:

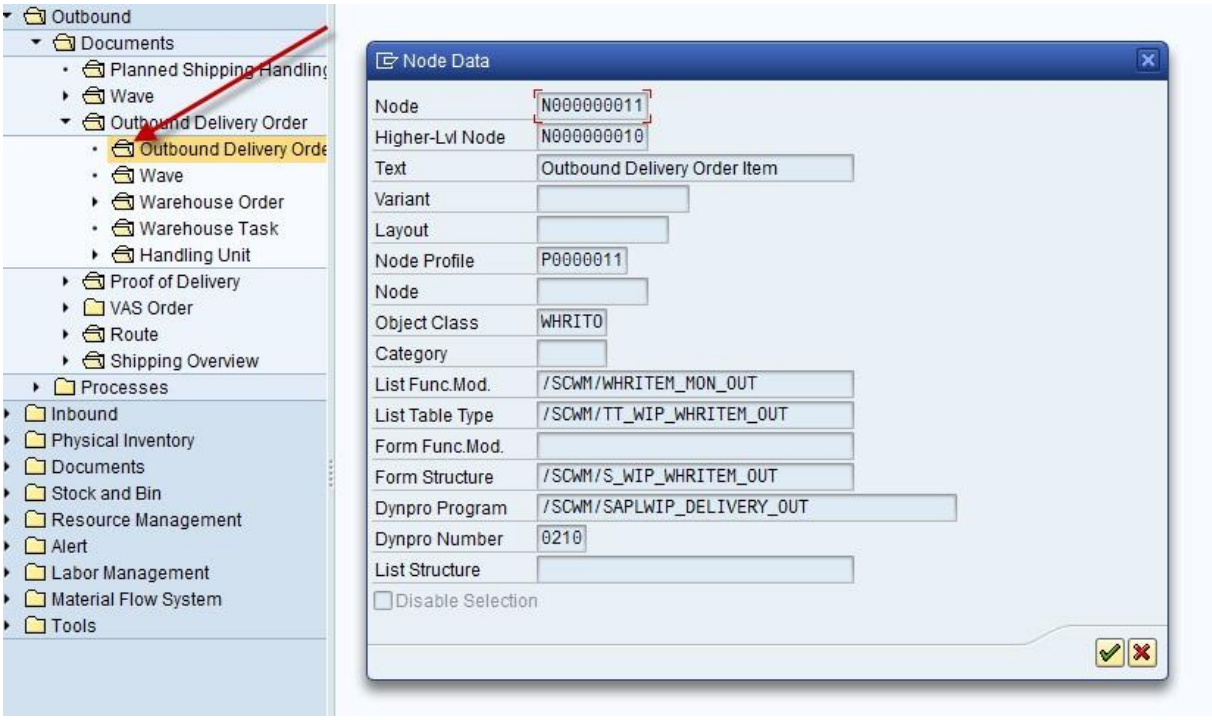


Figure: Node Details for Node "Outbound Delivery Order Item"

In our example, the node ID for the monitor node *Outbound Delivery Order Item* is *N000000011*. Keep the parent node ID *N000000010* in mind also.

4.2.3.2 Display All Node IDs of Monitor Node Hierarchy

The Customizing activity *Customize Monitor Tree* allows you to display all node IDs of a monitor node hierarchy. Press the icon to toggle node keys. The node keys of all tree nodes are displayed:

Figure: Display Node ID for Outbound Delivery Order Item

4.2.4 Copy Node to Customer Node

In the Customizing activity *Customize Monitor Tree* you can copy the SAP standard node to customer nodes. Press the icon *Open Monitor* and load the customer monitor *ZHTG* (on the right-hand side of the screen). Expand the node hierarchy below *Outbound Delivery Order*:

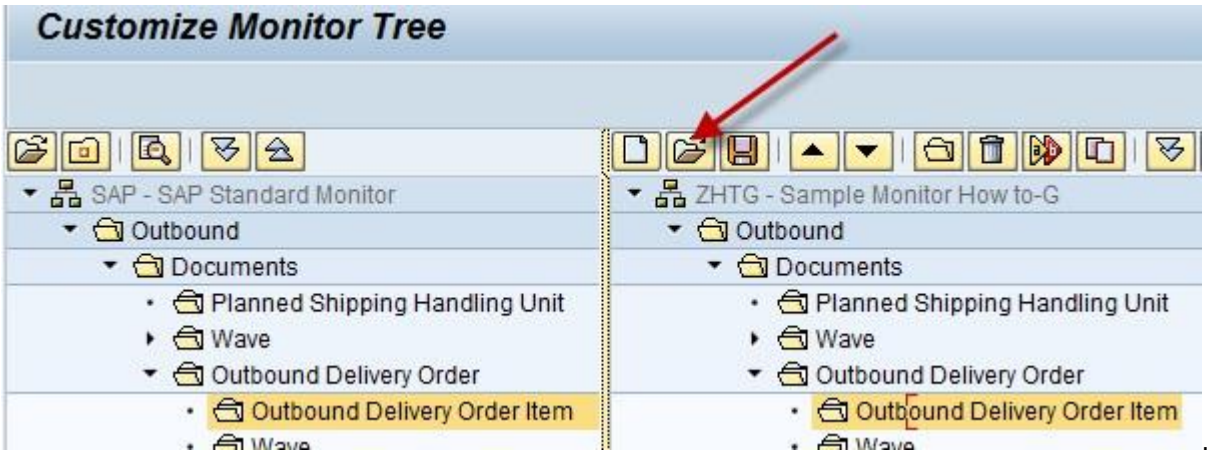


Figure: Load Customer Monitor

Mark the node *Outbound Delivery Order Item* and press the icon *Duplicate Node*. A copy of the node is inserted below the original node:

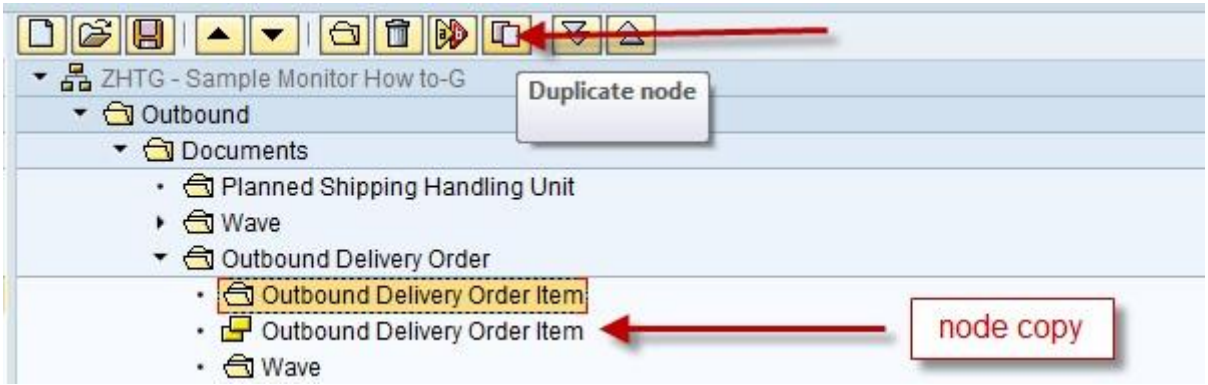


Figure: Copy Custom Node from Original Node

Then press the icon *Delete Node* to remove the parent-child relation of the original node:

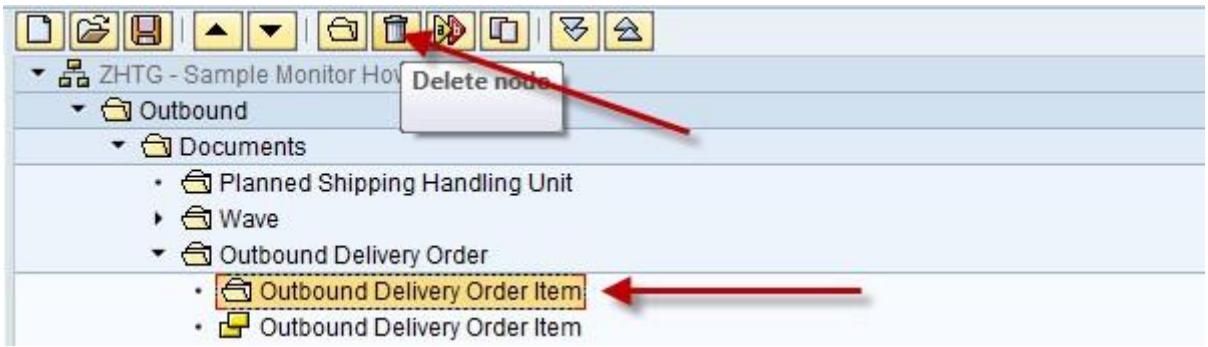


Figure: Remove Original Node from Customer Monitor

Mark the copied customer node. Press the icon *Rename Node* and enter the new node text *Outbound Delivery Ord. Item XT*:

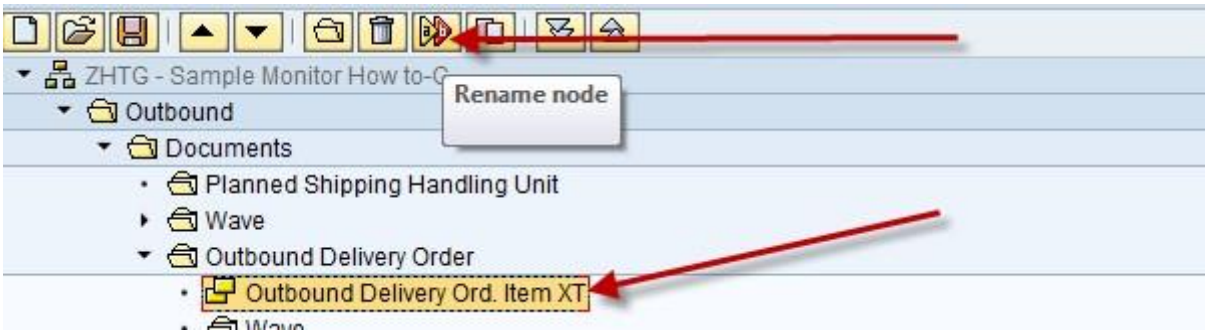


Figure: Rename Customer Node

Save your changes. The system automatically determines a new node ID for the customer node. Double-click on the customer node. Keep the customer node ID (in our example: *ZN00000003*) and the assigned profile. In our example the assigned node profile ID is *P0000011*.

4.2.5 Copy Node Profile

Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Define Nodes*:

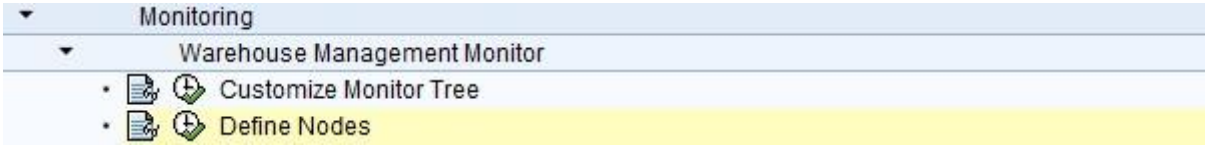


Figure: Customizing Activity for Changing Node Content like Node Profiles

On the following screen double-click on the folder *Define Node Profile*. Press the button *Position* and enter the profile ID. The profile entry is displayed. Mark the node profile entry and press the icon *Copy As* (`F6`). Enter the new node profile name *ZHTGODOI* and press `ENTER`. Profile data is copied from the profile *P0000011* to the new profile *ZHTGODOI*. Save your changes.

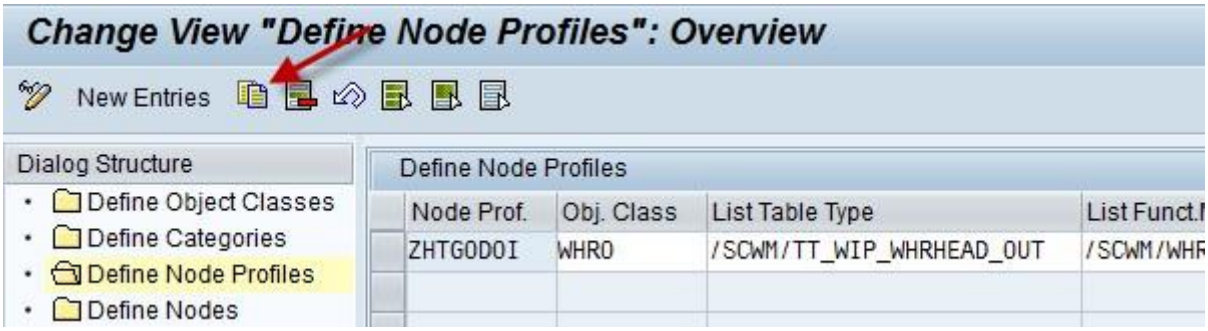


Figure: Copied Node Profile Data

The following profile data has to be exchanged against the enhanced customer data:

- List table type */SCWM/TT_WIP_WHRITEM_OUT*
- List function module */SCWM/WHRITEM_MON_OUT*
- Form structure */SCWM/S_WIP_WHRITEM_OUT*
- Dynpro program */SCWM/SAPLWIP_DELIVERY_OUT* (master program of function group */SCWM/WIP_DELIVERY_OUT*)
- Dynpro no *0210*

4.2.6 Copy and Enhance DDIC Structures

4.2.6.1 Enhance DDIC Structure used for List Display

The list table type */SCWM/TT_WIP_WHRITEM_OUT* uses the structure */SCWM/S_WIP_WHRITEM_OUT* as line type. Extending this structure with an additional field for the product detailed description would have an impact on all coding parts that use the original DDIC structure. Proceed as follows:

Create a new DDIC structure in the Data Dictionary (transaction SE11):

- Structure: *ZHTG_S_WIP_WHRITEM_OUT*
- Short Description: *Outbound Delivery Item in WIP Monitor XT*

On the tab strip *Components*, the following values have to be maintained:

Component	Typing Method	Component Type
.INCLUDE	Types	/SCWM/S_WIP_WHRITEM_OUT
MAKTX	Types	/SCWM/DE_UI_MAKTX

Tip

Use the menu path *Edit -> Include -> Insert* to add components of the original structure */SCWM/S_WIP_WHRITEM_OUT* as includes.

Activate and save your changes.

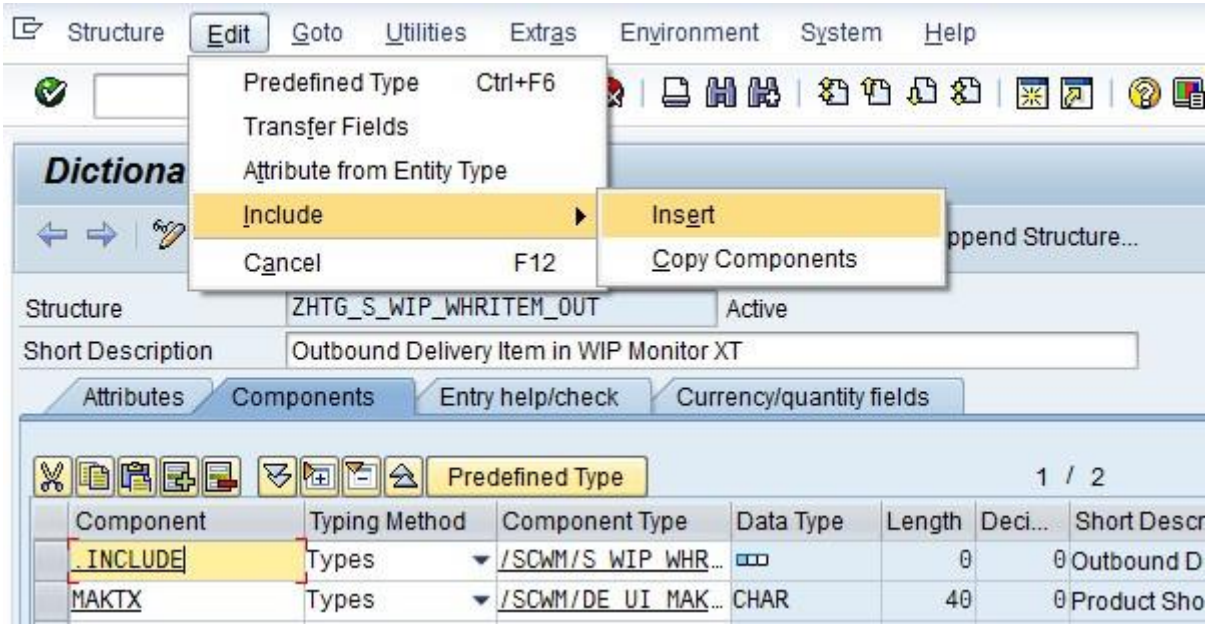


Figure: Enhanced DDIC Structure ZHTG_S_WIP_WHRITEM_OUT

4.2.6.2 Enhance DDIC Table Type used for List Display

The table type */SCWM/TT_WIP_WHRITEM_OUT* is copied to *ZHTG_TT_WIP_WHRITEM_OUT*. The line type */SCWM/S_WIP_WHRITEM_OUT* is exchanged against the enhanced structure *ZHTG_S_WIP_WHRITEM_OUT*. Activate and save your changes.

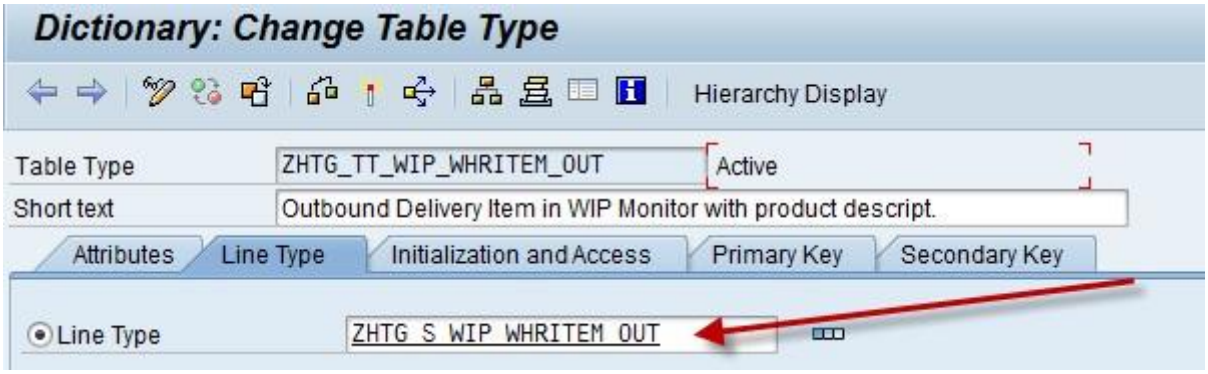


Figure: Enhanced DDIC Table Type ZHTG_TT_WIP_WHRITEM_OUT

4.2.6.3 Enhance DDIC Structure used for Form Display

Because the same DDIC structures are used for displaying data in form view and list view, you can exchange the form structure */SCWM/S_WIP_WHRITEM_OUT* against the new structure *ZHTG_S_WIP_WHRITEM_OUT*.

4.2.7 Create Wrapper Function Module

Start the Object Navigator (transaction *SE80*) and create a new wrapper function module:

- Function module: *ZHTG_WHRITEM_MON_OUT_XT*
- Function group: *ZHTG*
- Description: *Extended Delivery Item Selection for Monitor*

Maintain the following parameters for the function module interface. Copy the entries from the following tables into the fields of the function module interface: Tab strip *Import*:

Parameter Name	Typing	Associated Type	Default	Opt.	Short Text
IT_DATA_PARENT	TYPE	/SCWM/TT_WIP_WHRHEAD_OUT		X	Table consisting of (parent) entries selected for drill down
IV_CATEGORY	TYPE	/SCWM/DE_CATEGORY			Category
IV_LGNUM	TYPE	/SCWM/LGNUM			Warehouse Number/Warehouse Complex
IV_MODE	TYPE	/SCWM/DE_MON_FM_MODE	'1'		Execution mode for function modules in WM monitor
IV_VARIANT	TYPE	VARIANT		X	ABAP report variant

Note

The field *IV_MODE* determines how the function module is executed. The following values for *IV_MODE* are currently supported:
1 = normal; 2 = no selection screen; 3 = only choose selection variant; 4 = refresh

- Tab strip *Export*:

Parameter Name	Typing	Associated Type	Short Text

ET_DATA	TYPE	ZHTG_TT_WIP_WHRITEM_OUT	Outbound Delivery Item in WIP Monitor with product description
EV_RETURNCODE	TYPE	XFELD	'X': User cancelled selection
EV_VARIANT	TYPE	VARIANT	selected ABAP report variant

- Tab strip *Changing*:

Parameter Name	Typing	Associated Type	Opt.	Short Text
CT_FIELDCAT	TYPE	LVC_T_FCAT	X	Field Catalog for List Viewer Control
CT_RANGE	TYPE	RSDS_TRANGE	X	Table consisting of selection options of previous nodes

- Tab strip *Exceptions*:
    - Enter the class-based exception */SCWM/CX_MON_NOEXEC* Tab strip *Source code*:
    - Copy and paste the following lines of code

Source Coding of Sample Function Module ZHTG_WHRITEM_MON_OUT_XT

```
FUNCTION zhtg_whritem_mon_out_xt.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IT_DATA_PARENT) TYPE  /SCWM/TT_WIP_WHRHEAD_OUT
*"       OPTIONAL
*"     REFERENCE(IV_CATEGORY) TYPE  /SCWM/DE_CATEGORY
```

Source Coding of Sample Function Module ZHTG_WHRITEM_MON_OUT_XT

```
*"     REFERENCE(IV_LGNUM) TYPE  /SCWM/LGNUM
*"     REFERENCE(IV_MODE) TYPE  /SCWM/DE_MON_FM_MODE DEFAULT '1'
*"     REFERENCE(IV_VARIANT) TYPE  VARIANT OPTIONAL
*"  EXPORTING
*"     REFERENCE(ET_DATA) TYPE  ZHTG_TT_WIP_WHRITEM_OUT
*"     REFERENCE(EV_RETURNCODE) TYPE  XFELD
*"     REFERENCE(EV_VARIANT) TYPE  VARIANT
*"  CHANGING
*"     REFERENCE(CT_FIELDCAT) TYPE  LVC_T_FCAT OPTIONAL
*"     REFERENCE(CT_RANGE) TYPE  RSDS_TRANGE OPTIONAL
*"  RAISING
*"      /SCWM/CX_MON_NOEXEC
*"----------------------------------------------------------------------
  TYPES:
    BEGIN OF lty_s_prod_txt,
matid TYPE /sapapo/matid,
matnr TYPE /sapapo/matnr,
maktx TYPE /sapapo/maktx,
    END OF lty_s_prod_txt,
    lty_t_prod_txt TYPE SORTED TABLE OF lty_s_prod_txt
WITH NON-UNIQUE KEY matnr.
   DATA:     lt_data_tmp    TYPE
/scwm/tt_wip_whritem_out,     lt_prod_txt    TYPE
lty_t_prod_txt.   DATA:
    ls_prod_txt    TYPE lty_s_prod_txt.

  FIELD-SYMBOLS:
    <ls_data>      LIKE LINE OF et_data,
<ls_data_tmp>  LIKE LINE OF lt_data_tmp.

  CLEAR: et_data, ev_returncode, ev_variant.

* Wrapping standard function module
  CALL FUNCTION '/SCWM/WHRITEM_MON_OUT'
EXPORTING
      iv_lgnum       = iv_lgnum
iv_variant     = iv_variant
iv_mode        = iv_mode
it_data_parent = it_data_parent
IMPORTING
      et_data        = lt_data_tmp
ev_returncode  = ev_returncode
ev_variant     = ev_variant
CHANGING       ct_tab_range   =
ct_range.

* Get product short description
  SELECT m~matid m~matnr t~maktx
    FROM /sapapo/matkey AS m
      INNER JOIN /sapapo/mattxt AS t ON t~matid = m~matid
      INTO CORRESPONDING FIELDS OF TABLE lt_prod_txt
    FOR ALL ENTRIES IN lt_data_tmp
      WHERE m~matnr = lt_data_tmp-productno AND t~langu = sy-langu.

* Enhance result table with product short description
  LOOP AT lt_data_tmp ASSIGNING <ls_data_tmp>.
    APPEND INITIAL LINE TO et_data ASSIGNING <ls_data>.     MOVE-
CORRESPONDING <ls_data_tmp> TO <ls_data>.
```

Source Coding of Sample Function Module ZHTG_WHRITEM_MON_OUT_XT

```
    CLEAR ls_prod_txt.
    READ TABLE lt_prod_txt WITH KEY matnr = <ls_data_tmp>-productno
      INTO ls_prod_txt BINARY SEARCH.
    IF sy-subrc = 0.
      <ls_data>-maktx = ls_prod_txt-maktx.
    ELSE.
      CONTINUE.
    ENDIF.
  ENDLOOP.

  UNASSIGN: <ls_data>, <ls_data_tmp>.

ENDFUNCTION.
```

Tip

Use the function module */SCWM/MATERIAL_READ_MULTIPLE* of the function group */SCWM/MATERIAL_READ* for reading additional product attributes, instead of data selection by *INNER JOIN*. For the sake of simplicity this example uses direct data selection.

Important

Error messages must not be implemented using type „E". Use type „S" instead and add the ABAP phrase *DISPLAY LIKE 'E'*.

4.2.8 Copy and Adjust Screen (Form Layout)

Start the Object Navigator (transaction SE80) for the program */SCWM/SAPLWIP_DELIVERY_OUT* (the master program of the function group */SCWM/WIP_DELIVERY_OUT*). Open the folder *Screens* and select the screen *0210*. Copy the screen *0210* to the screen *0210* of the customer function group *ZHTG* (master program *SAPLZHTG*).

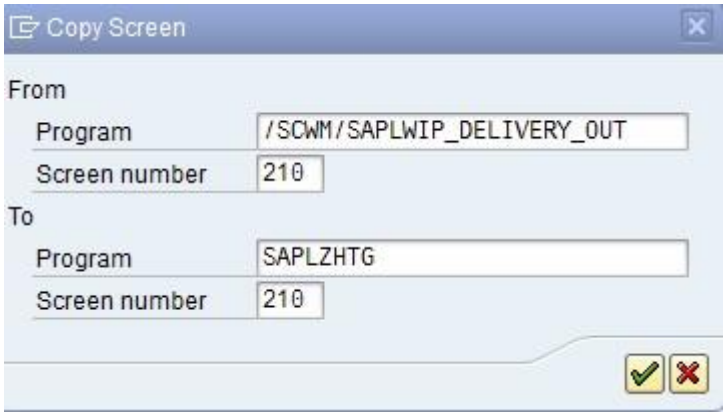


Figure: Copying Screen 0210 (Form View)

Navigate to the function group *ZHTG* and call the screen *0210*. Switch to layout mode. Enlarge the screen and move the screen elements below the field *Product*. Add the following screen elements:

El. Type	Name	Text	From Dict.
Text Field	ZHTG_S_WIP_WHRITEM_OUT-MAKTX	Product_Descr._	X
I/O Field	ZHTG_S_WIP_WHRITEM_OUT-MAKTX		X

Save and activate your changes.

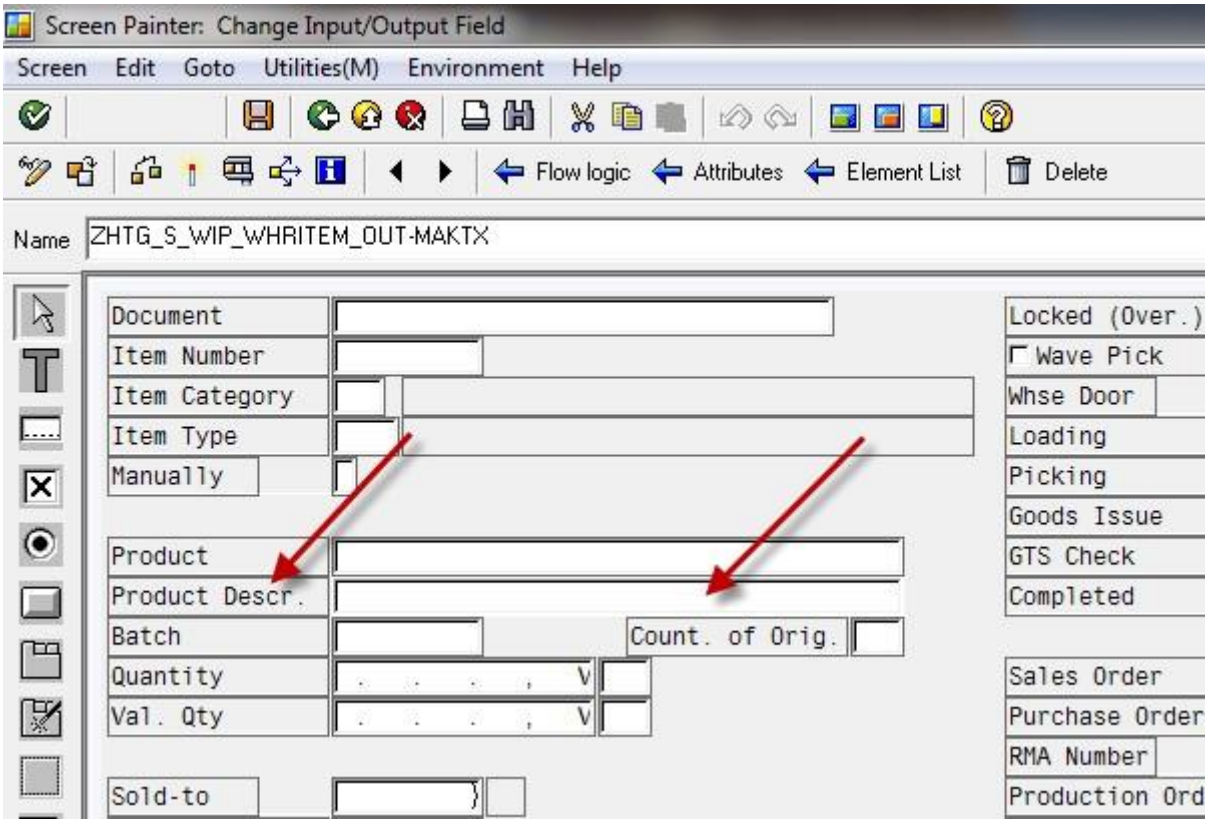


Figure: Form View Screen containing Label and Field for Product Short Description

4.2.9 Change Customizing Settings for Monitor Node

4.2.9.1 Exchange Node Profile of Customer Node

The new customized monitor node has the SAP standard profile assigned to it. Exchange this profile with the new profile *ZHTGODO*. Open the Customizing node *Extended Warehouse Management → Monitoring → Warehouse Management Monitor*. Execute the Customizing activity *Define Nodes*. Search for the custom node ID (in our example: *ZN00000003*). Set the value for the column *Node Prof.* to *ZHTGODOI.*

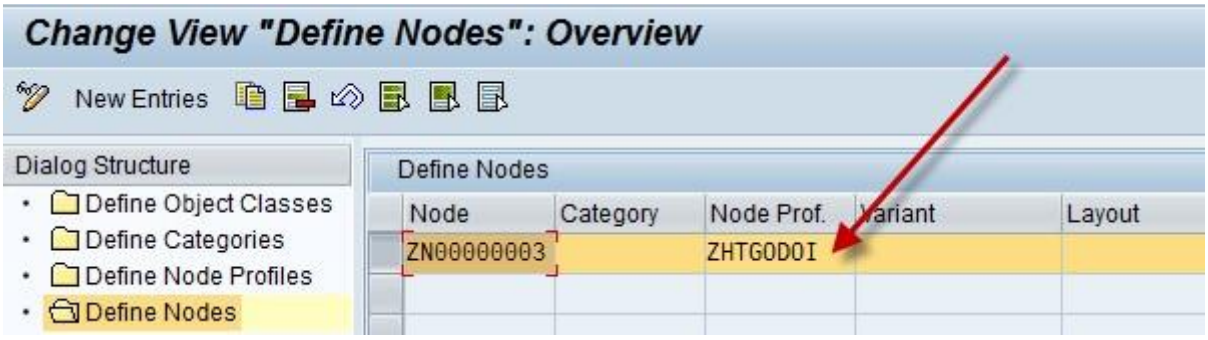


Figure: Node Profile Exchange for new Monitor Node ZHTGODOI01 Save

your changes.

4.2.9.2 Adjust Custom Node Profile

Double-click on the folder *Define Node Profiles*. Search for the node profile *ZHTGODOI*. Maintain the following values:

Column	Value

List Table Type	ZHTG_TT_WIP_WHRITEM_OUT
List Function Module	ZHTG_WHRITEM_MON_OUT_XT
Form Structure	ZHTG_S_WIP_WHRITEM_OUT
Dynpro Program	SAPLZHTG
Dynpro No.	0210
Text	Outbound Delivery Ord. Item XT
Presentation Text	ODO Item XT

Keep the other copied values and save your changes.

4.2.9.3 Optional: Maintain Hotspot Navigation to Outbound Delivery

Customizing settings for hotspot navigation for the column *Document* are described in chapter 7.

4.2.9.4 Check Customizing Changes

Execute the Customizing activity *Customize Monitor Tree* and load the customer monitor *ZHTG*. Expand the node hierarchy below the folder *Outbound* until the new custom node is displayed. Double-click on the folder *Outbound Delivery Ord. Item XT* to list the node attributes.

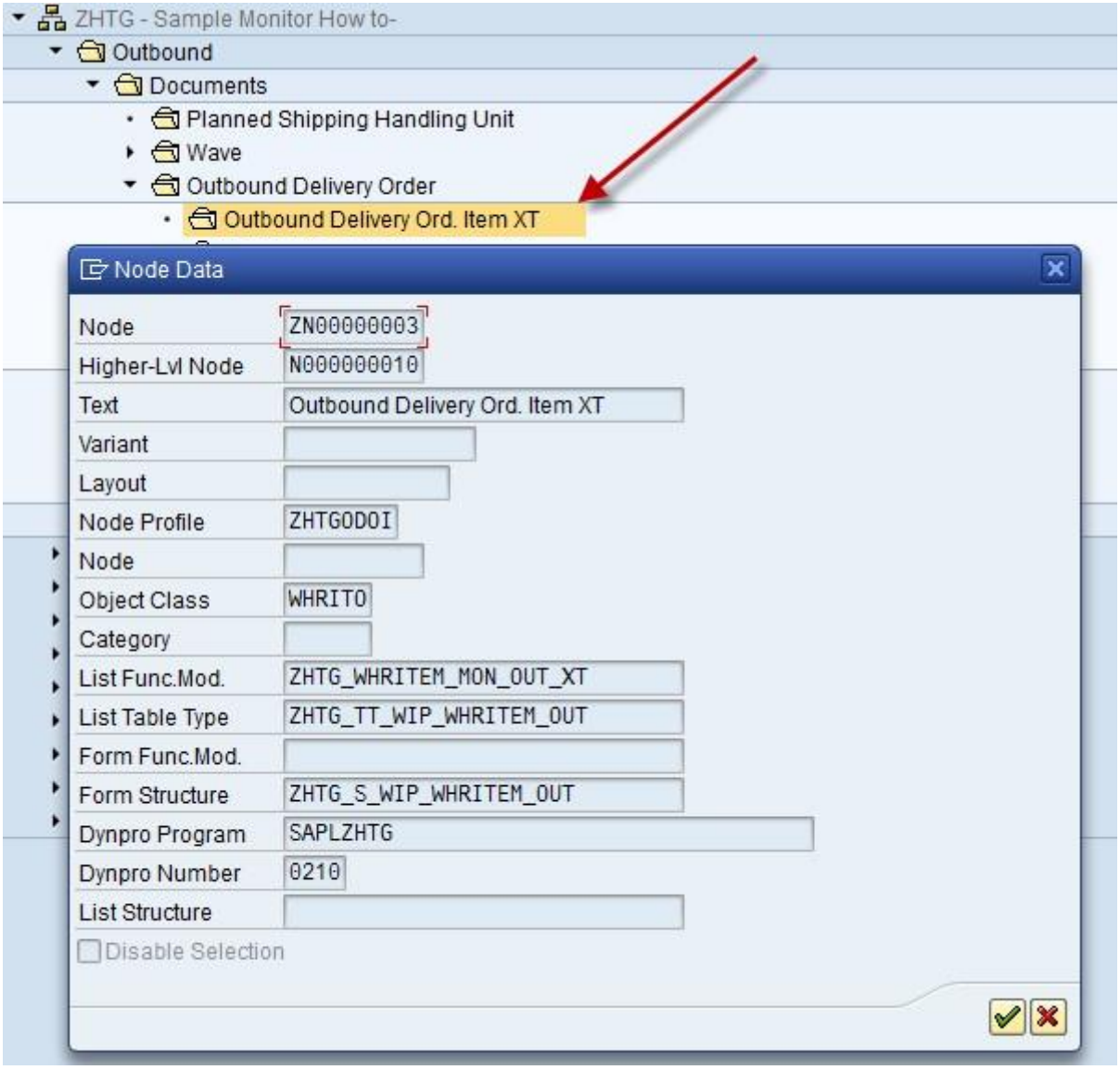


Figure: Display Node Data of Custom Node

4.2.10 Testing

Start the warehouse management monitor (transaction /SCWM/MON). Call the customer monitor *ZHTG*. Open the node *Outbound -> Documents -> Outbound Delivery Order.* Double-click on the folder *Outbound Delivery Ord. Item XT*.

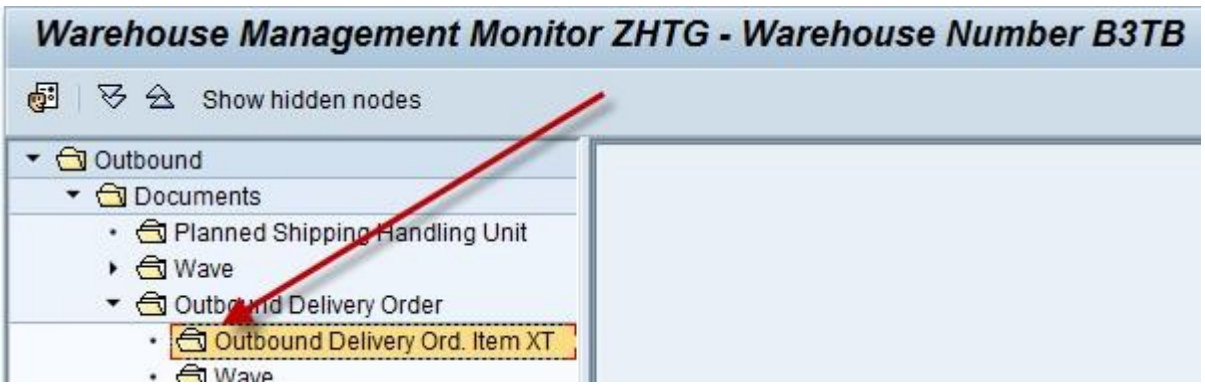


Figure: Selecting ODO Items using Enhanced Monitor Node

The selection screen appears. Maintain the appropriate entries and execute the search (`F8`). The result view contains the product short descriptions in one of the last columns of the result list. Define an ALV grid variant to change the position of the columns *Product* and *Product Short Descriptions*: Press the icon *Choose Layout* and select *Change Layout*.

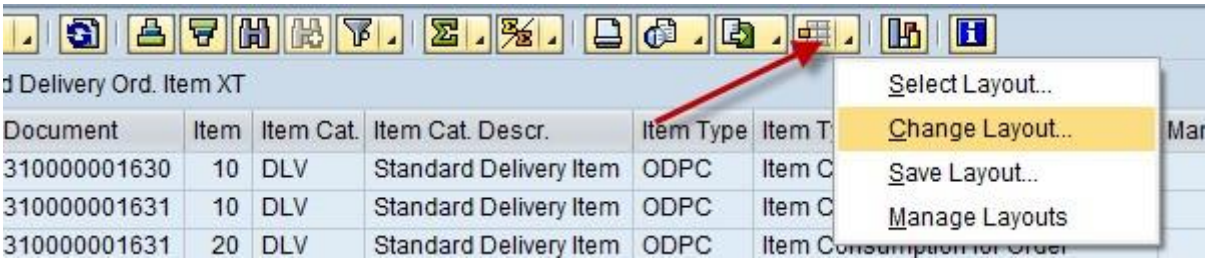


Figure: Change Layout of ALV Grid Result List

The dialog box *Change Layout* appears. Mark the entries *Product* and *Product Short Descriptions*. Use the button *Sel. Field(s) Up* (`F8`) to move the marked entries below the column *Item Type Description*. Choose the button *Save Layout As* and save your changes. Select the checkbox *Default* if you want your ALV layout to be used as the default settings for result list.

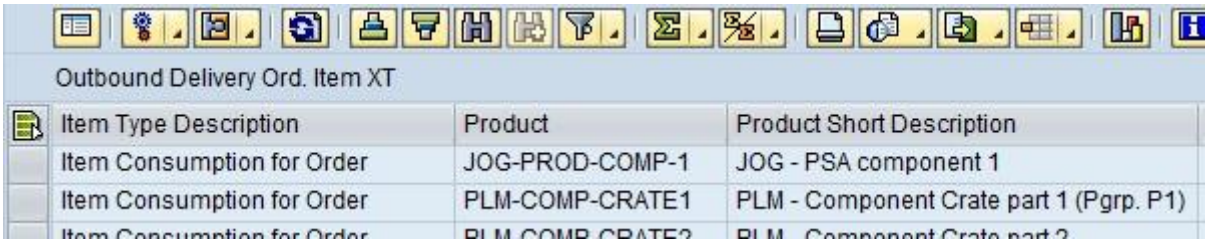


Figure: Changed order of New Columns "Product" and "Product Short Description"

Toggle to form view and check that the product short descriptions are also displayed.

Figure: Product Short Description Data (Form View)

Return to the hierarchy view. Double-click on the folder *Outbound Delivery Order*. Enter the selection criteria and execute the search. Mark the ODO entry in the result list and press the button *ODO Item XT*.

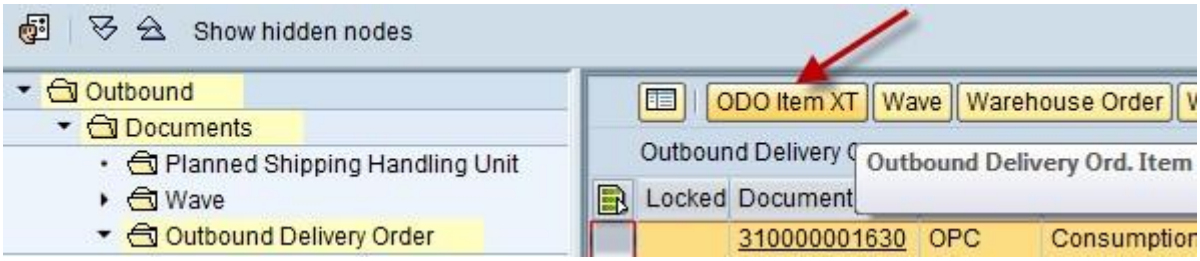


Figure: Navigation from Outbound Delivery Orders to Outbound Delivery Order Items

The outbound delivery order item data is displayed as desired.

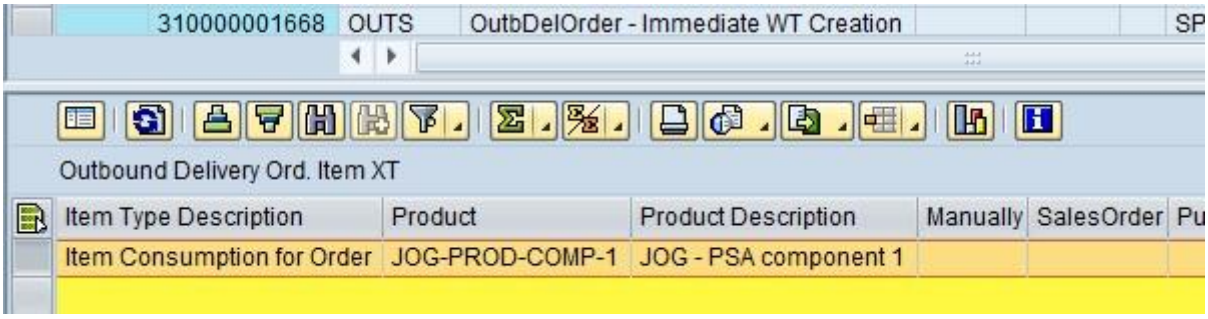


Figure: Product Short Description Data (List View)

4.3 Advanced Example: Extend Select Options for Deliveries and Display Results

In SAP standard it is not possible to search for inbound deliveries with the help of corresponding ERP document numbers. ERP numbers themselves are often used as selection criteria for first searches.

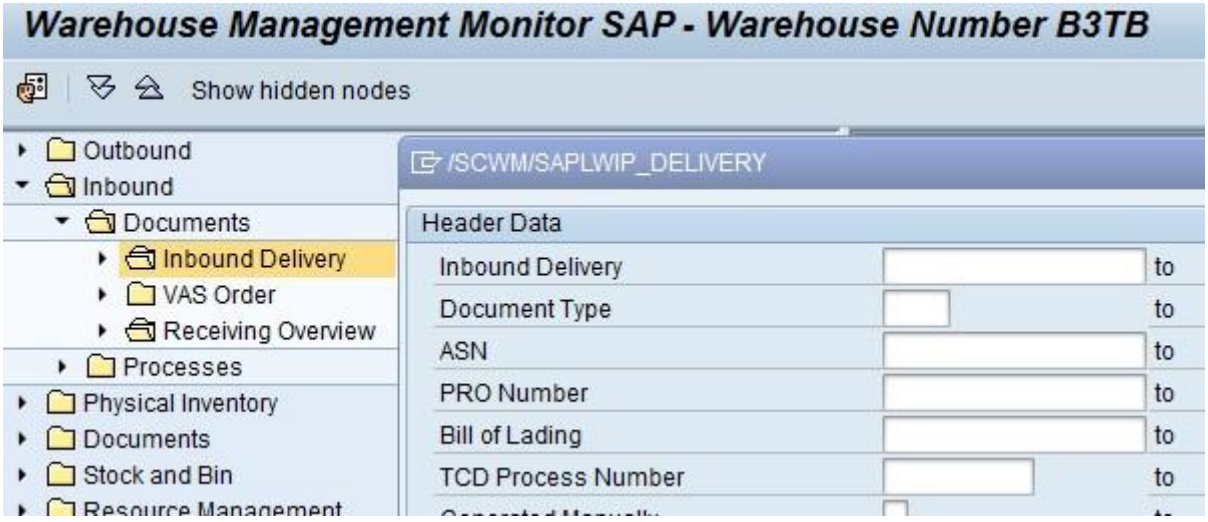


Figure: Selection Criteria for Inbound Deliveries without ERP Document Number as Select Option

The reuse of the customer monitor that was created for extending the outbound delivery item node allows reduction of overall efforts.

4.3.1 Solution Proposal

The customer solution shall fulfill the following requirements:

- Extend the selection screen with selection options for ERP documents
- Display ERP documents in the result list of inbound deliveries if the search has been executed using ERP documents
- Allow hotspot navigation to inbound deliveries

4.3.2 Required Steps

The following steps are required to display the product short description in the above-mentioned monitor node:

- Identify the node ID of the node *Inbound Delivery* and copy the node
- Identify the associated node profile and copy the profile
- Identify the DDIC structures used for displaying the item data and identify the function module that collects the order data
- Copy the DDIC structures into the customer namespace.

    - Append the field that shows the ERP document number
- Create a function module in the customer namespace that extends the functionality of the SAP standard function module used for collecting inbound delivery data.
    - Add coding that allows selection of inbound deliveries by ERP document ID Copy the screen that displays the data in form view into the customer function group.
    - Add a label and an output field for the ERP document number.
- Change the Customizing settings for the monitor node:
    - Exchange the function module, DDIC structures, screen program, and screen number in the copied node profile
    - Update the settings for the selected monitor node
- Test changes
    - Call the warehouse management monitor using the monitor hierarchy that contains the enhanced monitor node

4.3.2.1 Challenges

The extension of the selection criteria cannot be programmed with the help of wrapper function modules. Instead most of the logic that is implemented for the SAP standard function module *(/SCWM/WHRHEAD_MON*) has to be copied and adjusted.

4.3.3 Identification of Node Data

As described in the first example, use the Customizing activity *Customize Monitor* for getting the node data of the monitor node *Inbound Delivery*. In our example, the node ID for the monitor node *Inbound Delivery* is *N000000085*. Also note the ID of the parent node *C000000005*.

4.3.4 Copy Node to Customer Node

In the Customizing activity *Customize Monitor Tree* you can copy SAP standard nodes to customer nodes. Press the icon *Open monitor* and load the customer monitor *ZHTG* (on the right-hand side of the screen). Expand the node hierarchy below *Inbound Delivery*. Mark the node *Inbound Delivery* and press the icon *Duplicate node*. A copy of the node is inserted below the original node. Then press the icon *Delete node* to remove the parent-child relation of the original node.

Mark the copied customer node. Press the icon *Rename node* and enter the new node text *Inbound Delivery XT*. Save your changes. The system automatically determines a new node ID for the customer node. Double-click on the customer node. Note the customer node ID (in our example *ZN00000004*) and the assigned profile. In our example, the assigned node profile ID is *P0000085*.

4.3.5 Copy Node Profile

Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Define Nodes*. On the following screen double-click on the folder *Define Node Profiles*. Press the button *Position* and enter the profile ID. The profile entry is displayed. Mark the node profile entry and press the icon *Copy As* (`F6`). Enter the new node profile name *ZHTGID* and press `ENTER`. The profile data is copied from the profile *P0000085* to the new profile *ZHTGID*. Save your changes.

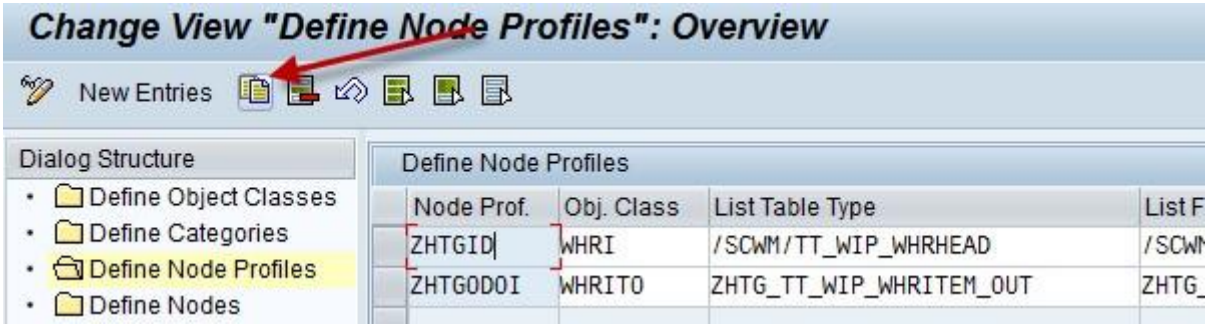


Figure: Copied Node Profile Data

The following profile data has to be exchanged against enhanced customer data:

- List table type */SCWM/TT_WIP_WHRHEAD*
- List function module */SCWM/WHRHEAD_MON*
- Form structure */SCWM/S_WIP_WHRHEAD*
- Dynpro program */SCWM/SAPLWIP_DELIVERY* (master program of function group */SCWM/WIP_DELIVERY_OUT*)
- Dynpro no *0110*

4.3.6 Copy and Enhance DDIC Structures

4.3.6.1 Enhance DDIC Structure used for List Display

The list table type */SCWM/TT_WIP_WHRHEAD* uses the structure */SCWM/S_WIP_WHRHEAD* as line type. Extending this structure with an additional field for the product detail description would have an impact on all coding parts that use the original DDIC structure. Proceed as follows:

Create a new DDIC structure in the Data Dictionary (transaction SE11):

- Structure: ZHTG_S_WIP_WHRHEAD
- Short Description: *Delivery Header in WIP Monitor XT*

On the tab strip *Components* maintain the following values:

Component	Typing Method	Component Type
.INCLUDE	Types	/SCWM/S_WIP_WHRHEAD
ERP_DOCNO	Types	/SCWM/SP_DOCNO_ERP

Tip

Use the menu path *Edit -> Include -> Insert* to add components of the original structure */SCWM/S_WIP_WHRHEAD* as includes

Activate and save your changes.

4.3.6.2 Enhance DDIC Table Type used for List Display

The table type */SCWM/TT_WIP_WHRHEAD* is copied to *ZHTG_TT_WIP_WHRHEAD*. The line type */SCWM/S_WIP_WHRHEAD* is exchanged against the enhanced structure *ZHTG_S_WIP_ WHRHEAD*.

- Short description: *Delivery Header in WIP Monitor XT*

Activate and save your changes.

4.3.6.3 Enhance DDIC Structure used for Form Display

The same DDIC structures are used for displaying the data in form view and in list view. Therefore, the form structure */SCWM/S_WIP_WHRHEAD* can be replaced with the new structure *ZHTG_S_WIP_ WHRHEAD*.

4.3.7 Create Function Module

As described above, it is not possible to wrap the function module */SCWM/WHRHEAD_MON* because selection options have to be enhanced. Start the Object Navigator (transaction SE80) and choose the function group *ZHTG*. Create the new function module *ZHTG_WHRHEAD_MON_XT*:

- Short text: *Delivery Header Selection for Monitor XT*

Copy the entries from the following tables into the fields of the function module interface:

- Tab strip *Import*:

Parameter Name	Typing	Associated Type	Default	Opt.	Short Text
IV_LGNUM	TYPE	/SCWM/LGNUM			Warehouse Number/Warehouse Complex
IV_MODE	TYPE	/SCWM/DE_MON_FM_MODE	'1'		Execution mode for function modules in WM monitor
IV_VARIANT	TYPE	VARIANT		X	ABAP report variant

Note

The field *IV_MODE* determines how the function module is executed. The following values for IV_MODE are currently supported:
1 = normal; 2 = no selection screen; 3 = only choose selection variant; 4 = refresh

- Tab strip *Export*:

Parameter Name	Typing	Associated Type	Short Text
ET_DATA	TYPE	ZHTG_TT_WIP_WHRHEAD	Delivery Header in WIP Monitor XT
EV_RETURNCODE	TYPE	XFELD	'X': User cancelled selection
EV_VARIANT	TYPE	VARIANT	selected ABAP report variant

- Tab strip *Changing*:

Parameter Name	Typing	Associated Type	Opt.	Short Text
CT_FIELDCAT	TYPE	LVC_T_FCAT	X	Field Catalog for List Viewer Control
CT_TAB_RANGE	TYPE	RSDS_TRANGE	X	Table consisting of selection options of previous nodes

- Tab strip *Exceptions*:
  - Enter the class-based exception /SCWM/CX_MON_NOEXEC Tab

strip *Source code*:
  - Copy and paste lines of code from the appendix (chapter 8.1.5)

Save your changes.

Create the following new includes in the function group *ZHTG*:

- *LZHTGTOP* Top include
- *LZHTGP01* Local Helper Class (Definition)
- *LZHTGP02* Local Helper Class (Implementation)

Copy the source code from the appendix into the includes and master program of the function group *ZHTG*.

- Master program *SAPLZHTG* appendix chapter 8.1.1 Include *LZHTGTOP* appendix chapter 8.1.2
- Include *LZHTGP01* appendix chapter 8.1.3
- Include *LZHTGP02* appendix chapter 8.1.4

Tip

The include containing the definition of local classes has to be included in the TOP include of the function group. The include containing the implementation of local classes has to be included in the master report of the function group.

Save and activate your changes. Then maintain the text symbols and selection texts as described in appendix chapter 8.1.6. Save and activate your changes again.

Notes on implementation:
The functional logic of this function module does not use subroutines (*FORM xyz*) but instead uses methods of local classes (ABAP OO). The main reasons for this are:

- Prevention of unwanted reuse of functionality as is possible for subroutines (dynamic PERFORM calls) by proper encapsulation
- Better testability of local class methods by ABAP unit tests, reducing maintenance efforts

4.3.7.1 Comments on Implementation

- The implementation of the function module used the SAP standard function module */SCWM/ WHRHEAD_MON* as a template
  - Subroutines are replaced by instance methods of the local helper class
- Adding ERP documents as selection criteria required changes in the following parts of the coding:
  - Include *LZHTGTOP*: Additional selection option *so_erpid* ○ Include *LZHTGP02*:
    - Method *add_sel_crit_common*: Add additional selection criteria for ERP document ID
  - Function module *ZHTG_WHRHEAD_MON_XT*:
    - Use reference documents in header data for determining ERP document number

4.3.8 Copy and Adjust Screen (Form Layout)

Start the object navigator (transaction SE80) for the program */SCWM/SAPLWIP_DELIVERY* (the master program of the function group */SCWM/WIP_DELIVERY*). Open the folder *Screens* and select screen 0110. Copy screen 0110 to screen 0110 of the customer function group *ZHTG* (master program *SAPLZHTG*). Navigate to the function group *ZHTG* and call screen 0110. Switch to layout mode. Enlarge the screen and move the screen elements below the field *Manually*. Add the following screen elements:

El. Type	Name	Text	From Dict.
Text Field	ZHTG_S_WIP_WHRHEAD-ERP_DOCNO	ERP Document	X
I/O Field	ZHTG_S_WIP_WHRHEAD-ERP_DOCNO		X

Save and activate your changes.

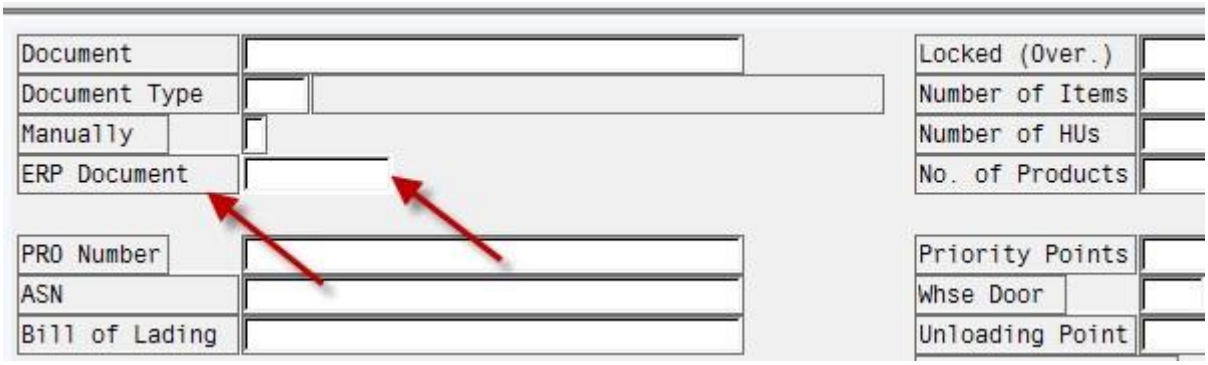


Figure: Form View Screen containing Label and Field for ERP Document

4.3.9 Change Customizing Settings for Monitor Node

4.3.9.1 Exchange Node Profile of Customer Node

The customized new monitor node has the SAP standard profile assigned to it. Exchange this profile with the new profile *ZHTGID*. Open the Customizing node *Extended Warehouse Management → Monitoring → Warehouse Management Monitor"*. Execute the Customizing activity *Define Nodes*. Search for the custom node ID (in our example *ZN00000004*). Set the value for the column *Node Prof.* to *ZHTGID.* Save your changes.

4.3.9.2 Adjust Customer Node Profile ZHTGID

Double-click on the folder *Define Node profiles*. Search for the node profile *ZHTGID*. Maintain the following values:

Column	Value
List Table Type	ZHTG_TT_WIP_WHRHEAD
List Function Module	ZHTG_WHRHEAD_MON_XT
Form Structure	ZHTG_S_WIP_WHRHEAD
Dynpro Program	SAPLZHTG
Dynpro No.	0110
Text	Inbound Delivery XT
Presentation Text	Inb. Deliv. XT

Keep other copied values and save your changes.

4.3.9.3 Optional: Maintain Hotspot Navigation to Inbound Delivery

The Customizing settings for hotspot navigation for the column *Document* are described in chapter 7.

4.3.9.4 Check Customizing Changes

Execute the Customizing activity *Customize Monitor Tree* and load the customer monitor *ZHTG*. Expand the node hierarchy below the folder *Inbound* until the new custom node is displayed. Double-click on the folder *Inbound Delivery* to see the node attributes:

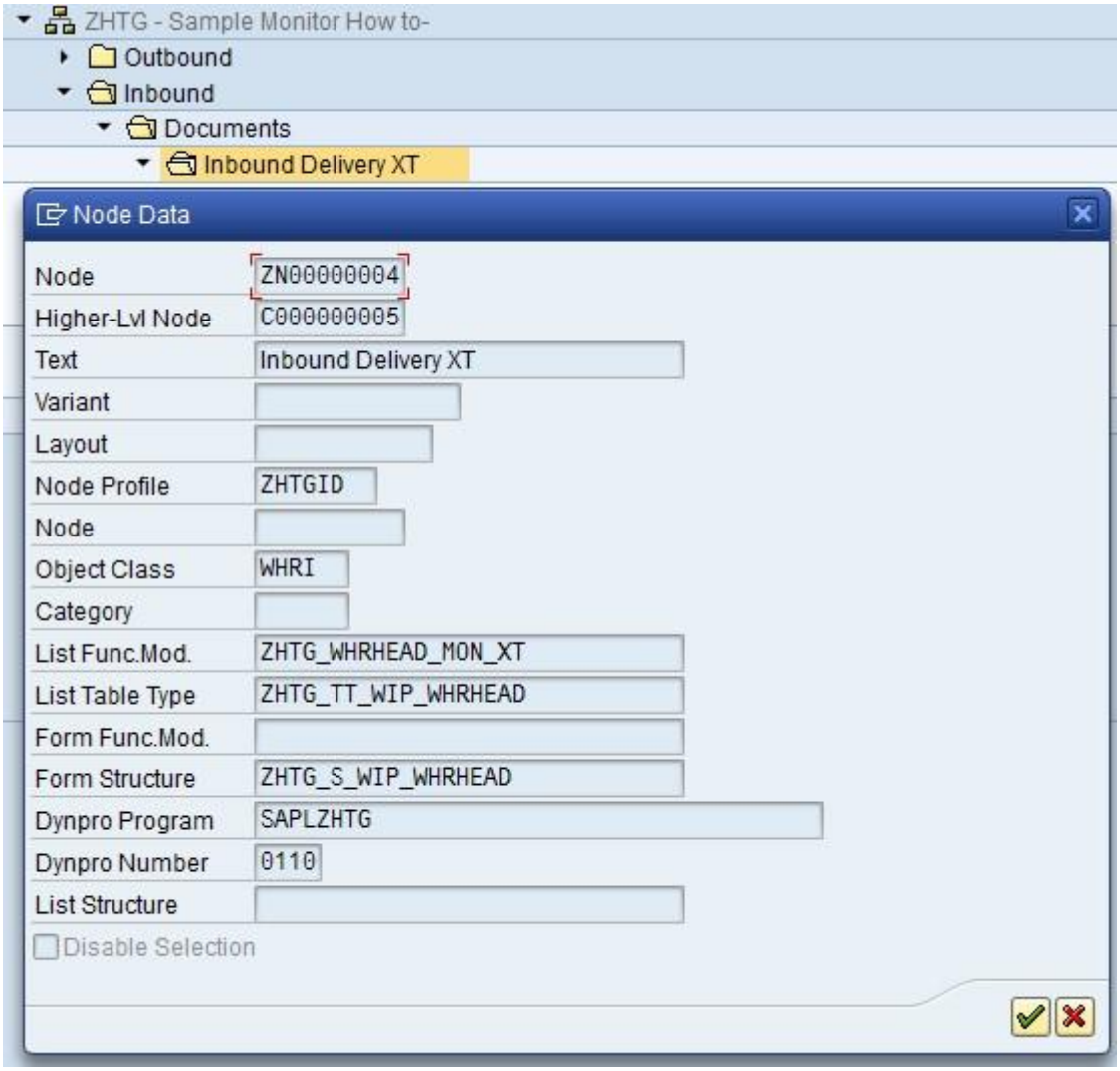


Figure: Display Node Data of Custom Node for Inbound Deliveries

4.3.10 Testing

Start the warehouse management monitor (transaction /SCWM/MON). Call the customer monitor *ZHTG*. Open the node *Inbound -> Documents -> Inbound Delivery*. Double-click on the folder *Inbound Delivery*. A selection screen appears that contains selection options for ERP documents.

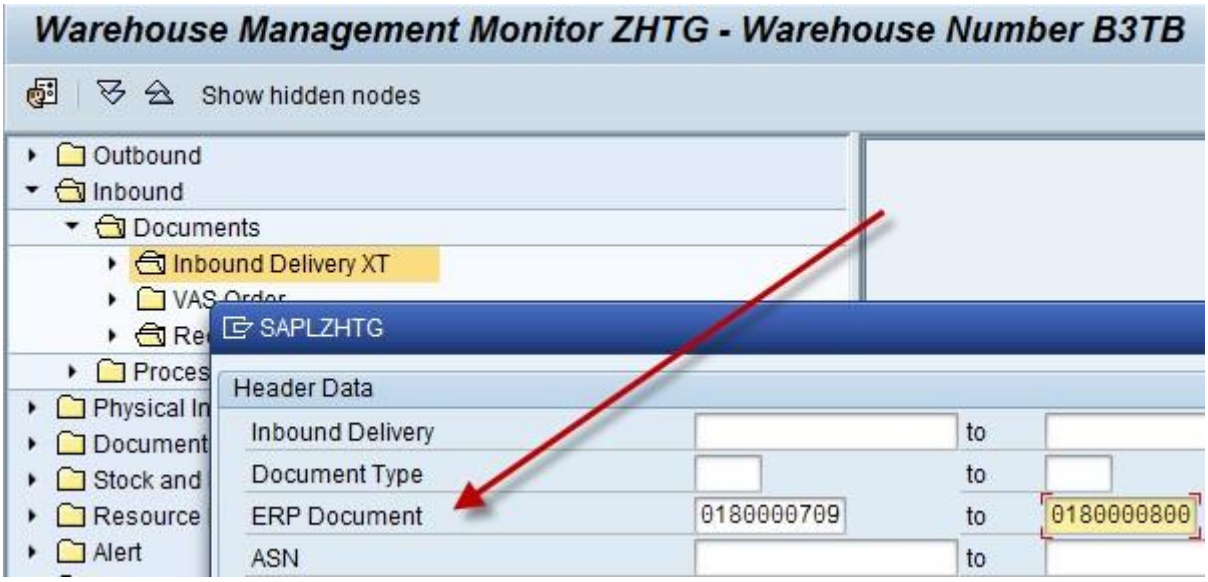


Figure: Selecting Inbound Deliveries using ERP Document Numbers

Enter values for ERP documents and execute the selection. Check that the ERP document number is displayed in the column *ERP Doc.* of the result list.

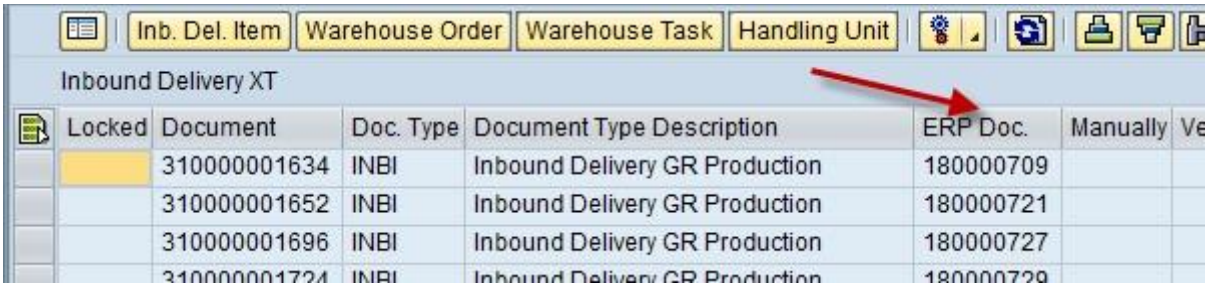


Figure: ERP Document Data (List View)

Mark the entries in the result list. Toggle to form view and check that the ERP document numbers are also displayed.

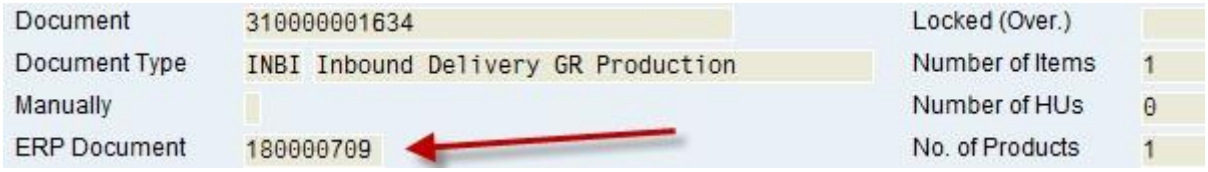


Figure: ERP Document Data (Form View)

Return to hierarchy view. Double-click on the folder *Outbound Delivery Order*. Enter selection criteria and execute the search. Mark the ODO entry in the result list and press the button *ODO Item XT*.

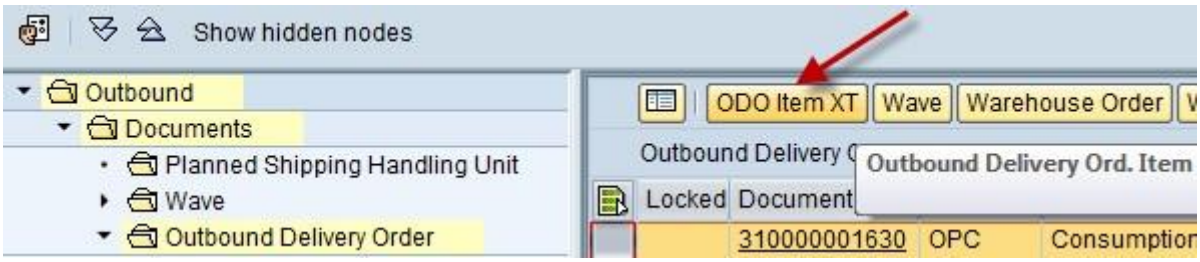


Figure: Navigation from Outbound Delivery Orders to Outbound Delivery Order Items

The outbound delivery order item data is displayed as desired:

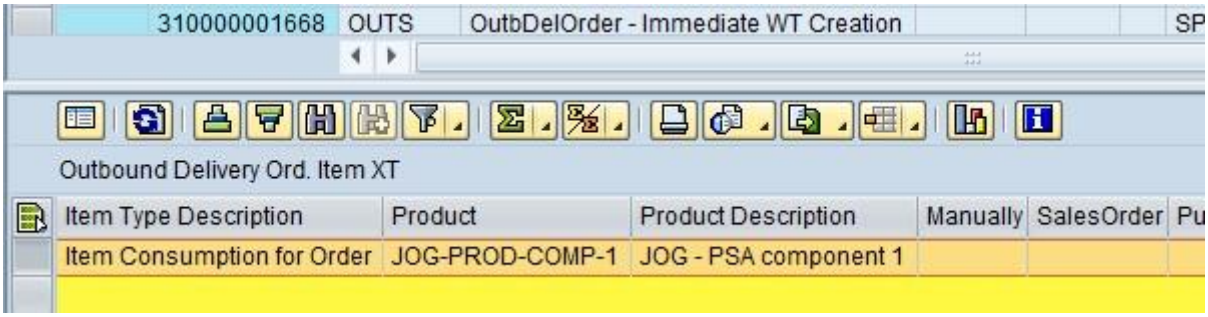


Figure: Product Short Description Data (List View)

5. Add New Methods to Monitor

As mentioned in Chapter 1, you can specify particular methods that can be started from the monitor for specific objects. Like the function modules in the previous chapters, the methods must also follow several guidelines.

5.1 Example: New Method for Unassigning Outbound Delivery Order Items from Wave

Outbound delivery order items can be assigned to waves for collective processing and release. This functionality is also available for outbound delivery items that are displayed in the result list of the corresponding monitor node.

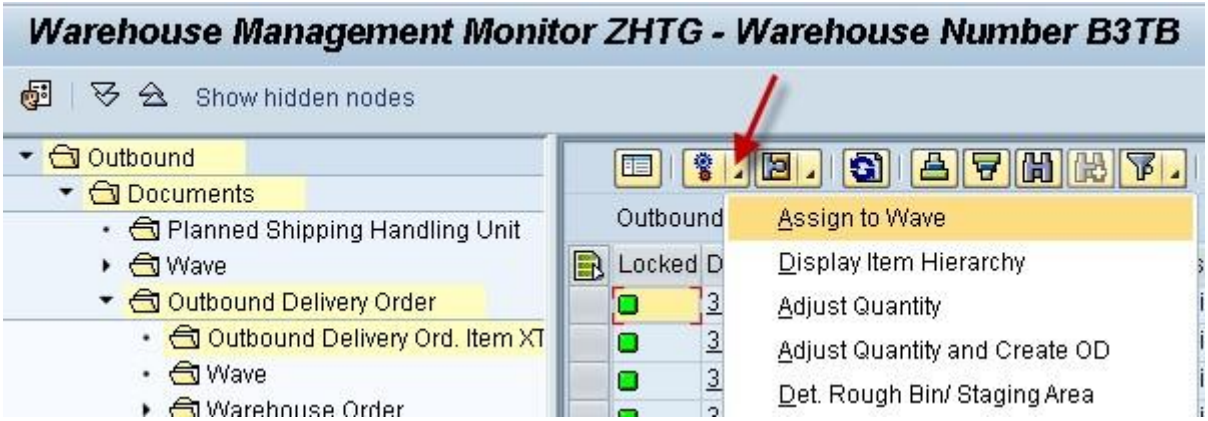


Figure: Method for Assigning Marked Outbound Delivery Order Items to Waves

Unfortunately there is no similar method available that allows unassigning of items from waves. The following chapters describe how to implement this new method.

5.1.1 Solution Proposal

The customer solution shall fulfill the following requirements:

- Provide a method *Unassign from Wave* for outbound delivery order items in the result view
  - Marked items that are assigned to waves shall be unassigned after the method is executed

5.1.2 Required Steps

As usual implementation is done for the monitor node *Outbound Delivery Ord. Item XT* of the custom monitor *ZHTG*. This prevents changing the SAP standard functionality in an unwanted way.

The following steps are required to implement a method for unassigning items from waves:

- Create a function module that is used in the new object class method
- Identify the object class of the node "*Outbound Delivery Ord. Item XT*"
- Define a new custom object class
- Copy the required object class methods from the object class copying template to the custom object class
- Add a new object class method for unassigning waves to the custom object class
- Change Customizing settings for the monitor node *Outbound Delivery Ord. Item XT*
    - Exchange the object class with the custom object class
- Test your changes

5.1.3 Prerequisites for Function Modules used in Object Class Methods

Object class methods call special function modules for processing. The following prerequisites have to be fulfilled by these function modules.

5.1.3.1 Interface of Function Modules used in Object Class Methods

The interface of these function modules has to contain the following parameters:

Parameter	Type	Associated Type	Description
IV_LGNUM	Imp.	/SCWM/LGNUM	Warehouse number/complex
IT_DATA	Imp.	STANDARD TABLE	Table consisting of selection of select entries in the monitor

Important

The table *IT_DATA* contains entries that have been marked in the ALV of the monitor framework. The method should be applied for these entries. Since the structure of these entries might be different for the same object class, the parameter has to use the generic type standard table.

5.1.3.2 Important Processing Steps

In the function module, the application programmer must exercise care with regard to the data passed to the function module. One of the first steps should be to move the data passed by the monitor framework into an internal table that can be processed by the method that is to be executed. Since the data type that is passed to the method is not fixed, the parameter *IT_DATA* has to use the type *STANDARD TABLE*. The conversion of the data and further processing is the responsibility of the application programmer. For example, the following points should be kept in mind:

- Authorization checks
- Since the function module cannot be sure that the data displayed in the monitor is the actual data stored in the database, the method should deal with this (for example, by locking the entry, comparing with the database, and reacting appropriately).
- The table *IT_DATA* might be empty. In this case, the function module has to check whether an error message should be displayed.
- Check the number of lines of the table *IT_DATA* in case the object class method can only be executed for one marked result list entry. Use the ABAP statement *lines(IT_DATA)* for determining the number of marked entries.

5.1.4 Sample Function Module ZHTG_WAVEMON_UNASSIGN

Start transaction SE80 and navigate to the function group *ZHTG*. Create the function module *ZHTG_ WAVEMON_UNASSIGN*.

- Short text: *Cancel Wave Item Assignment (Object Class Method Usage)* Copy the entries from the following tables into the fields of the function module interface:
- Tab strip *Import*:

Parameter Name	Typing	Associated Type	Default	Opt.	Short Text
IV_LGNUM	TYPE	/SCWM/LGNUM			Warehouse number/complex
IT_DATA	TYPE	STANDARD TABLE			Table containing selected ODO item entries

- Tab strip *Source code*:
  - Copy the following lines of code:

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN

```
FUNCTION zhtg_wavemon_unassign.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IV_LGNUM) TYPE  /SCWM/LGNUM
*"     REFERENCE(IT_DATA) TYPE  STANDARD TABLE
*"----------------------------------------------------------------------
-    DATA:
    lt_bapiret           TYPE bapiret2_t,
    lt_odo_item_xt                    TYPE   zhtg_tt_wip_whritem_out,
lt_wave_unass        TYPE /scwm/tt_wave_itm.
  DATA:
    ls_display_profile   TYPE bal_s_prof,     ls_log
TYPE bal_s_log,     ls_odo_item_xt       TYPE
zhtg_s_wip_whritem_out.
  DATA:     lv_lines_w_wave      TYPE i,
lv_loghandle         TYPE balloghndl,
lv_selected_lines    TYPE i.
  DATA:
    lo_log               TYPE REF TO /scwm/cl_log.

  CONSTANTS:
    lc_doccat_pdo        TYPE /scwm/de_doccat VALUE 'PDO'.

  FIELD-SYMBOLS:
    <ls_data> TYPE any.

  CREATE OBJECT lo_log.

**********************************************************************
```

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN

```
*   Check that entries have been selected from result list
*********************************************************************
lv_selected_lines = lines( it_data ).
  IF lv_selected_lines < 1.
*   No data transferred -> stop execution
    MESSAGE s050(/scwm/wave) DISPLAY LIKE 'E'. *
No delivery items were transferred
    RETURN.
  ENDIF.
*********************************************************************
*   Move data into table which can be used for further processing       *
*********************************************************************
  LOOP AT it_data ASSIGNING <ls_data>.
    MOVE-CORRESPONDING <ls_data> TO ls_odo_item_xt.
    APPEND ls_odo_item_xt TO lt_odo_item_xt.
  ENDLOOP.
*   Determine wave items of waves in status 'initial' (I) or 'hold' (H)
*   that belong to selected outbound delivery order items
  SELECT m~lgnum m~wave m~wave_itm
    FROM /scwm/waveitm AS m
      INNER JOIN /scwm/wavehdr AS t ON
        ( t~wave = m~wave AND t~lgnum = m~lgnum )
      INTO CORRESPONDING FIELDS OF TABLE lt_wave_unass
    FOR ALL ENTRIES IN lt_odo_item_xt
      WHERE m~lgnum = iv_lgnum AND
m~docno = lt_odo_item_xt-docno_h AND
m~itemno = lt_odo_item_xt-itemno AND
( t~status = wmegc_stwave_initial OR
t~status = wmegc_stwave_hold ).
   lv_lines_w_wave = lines( lt_wave_unass
).

  IF lv_lines_w_wave = 0.
*     No data that contains wave -> stop processing
    MESSAGE s008(/scwm/wave) DISPLAY LIKE 'E'. *
No items found for removal
    RETURN.
  ELSE.
    IF lv_lines_w_wave < lv_selected_lines.                "#EC NEEDED *
Info: Not all selected entries were assigned to waves and could
*     be unassigned -> add custom info message
    ENDIF.
  ENDIF.
*********************************************************************
*     Process data                                                     *
*********************************************************************
CALL FUNCTION '/SCWM/WAVE_ITEMS_UNASSIGN_EXT'     EXPORTING
      iv_lgnum        = iv_lgnum
iv_rdoccat      = lc_doccat_pdo
it_wave_unass   = lt_wave_unass
iv_update_task = 'X'
iv_commit_work = 'X'     IMPORTING
et_bapiret      = lt_bapiret.

  IF NOT lt_bapiret[] IS INITIAL.
```

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN

```
*     Handle messages that occurred during function processing
lo_log->add_log( it_prot = lt_bapiret ).     ls_log-extnumber = 1.
ls_log-object = wmegc_apl_object_wme.     ls_log-subobject =
wmegc_apl_subob_gen.

    lo_log->create_log(
EXPORTING
        is_log       = ls_log
IMPORTING         ev_loghandle =
lv_loghandle ).
     lo_log->convert_bapiret2applog(
).

*     Get profile for popup application log
    CALL FUNCTION 'BAL_DSP_PROFILE_POPUP_GET'
IMPORTING         e_s_display_profile =
ls_display_profile.
     ls_display_profile-use_grid =
'X'.

    TRY.         lo_log-
>display_log(
EXPORTING
            iv_loghandle       = lv_loghandle
is_display_profile = ls_display_profile ).

      CATCH /scwm/cx_basics.                          "#EC NO_HANDLER
    ENDTRY.
  ENDIF.

ENDFUNCTION.
```

Remarks:

- The main functional logic was taken from the SAP standard function module */SCWM/ WAVEMON_UNASSIGN*.
    - The selected outbound delivery order item data (*IT_DATA*) does not contain the necessary information about the corresponding wave items. Therefore additional steps for selecting the wave item data are required.
    - Error messages must not be implemented using type „E". Use type „S" instead and add the ABAP phrase *DISPLAY LIKE 'E'*.
- The import parameter *IT_DATA* uses the generic type (STANDARD TABLE). An additional check for correct typing of *IT_DATA* using the methods of the class *CL_ABAP_TYPEDESCR* is implemented in the function module *ZHTG_WAVEMON_UNASSIGN_XT* (see appendix 8.1.6)
- The database table */SCWM/WAVEHDR* is not buffered; therefore, *JOIN* execution does not cause performance issues.
- Removal of wave assignment is only implemented for waves with the status *Initial* or *Locked*. The setting of the indicator *Wave Assignment Also Possible After Wave Release* in the attributes of the wave template is not taken into account.

5.1.5 Define New Custom Object Class ZHTGOI

If new functionality is to be available for all nodes of the monitor tree, assign the corresponding object class method to the existing object classes to which the monitor nodes are assigned. In our example, existing object classes are not to be changed and enhanced with the new custom method. Therefore it is necessary to create a new custom object class.

Object classes and object class methods are created with the help of Customizing for the warehouse management monitor. In SAP Customizing go to the activity *Extended Warehouse Management → Monitoring → Warehouse Management Monitor → Define Nodes* to create the corresponding Customizing entries.

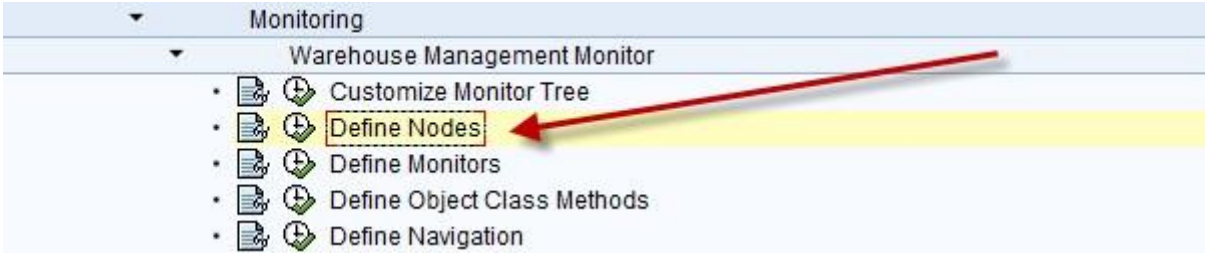


Figure: Customizing Activity for Creating Monitor Nodes

Execute the Customizing activity *Define Nodes*. Mark the folder *Define Object Classes* and press the button *New Entries*.

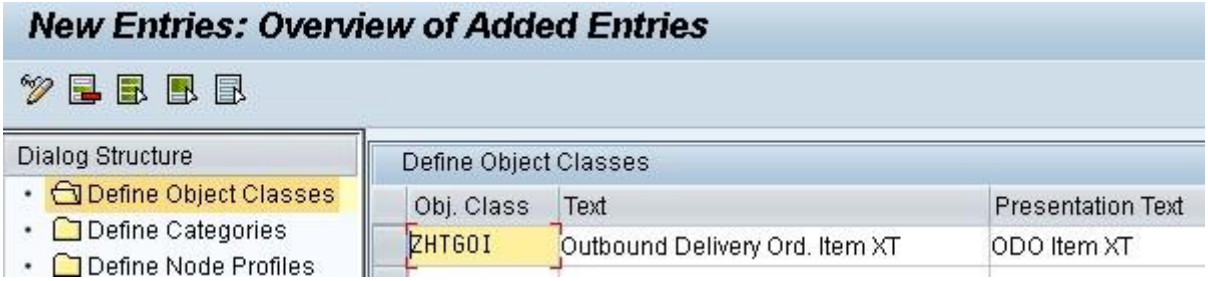


Figure: Custom Object Class for New Object Class Method

Maintain the entries as described in the following table and save your changes.

Obj. Class	Text	Presentation Text
ZHTGOI	Outbound Delivery Ord. Item XT	ODO Item XT

5.1.6 Maintain Object Class Methods for Custom Object Class ZHTGOI

To make a new object class method accessible in the monitor framework, the following tables have to be maintained in Customizing for EWM. The corresponding views can be found under the Customizing node *Extended Warehouse Management → Monitoring → Warehouse Management Monitor*.

Figure: Customizing Activity for Creating Object Class Methods

Execute the Customizing activity *Define Object Class Methods*. On the following screen double-click on the folder *Define Methods*. Press the button *New Entries*:

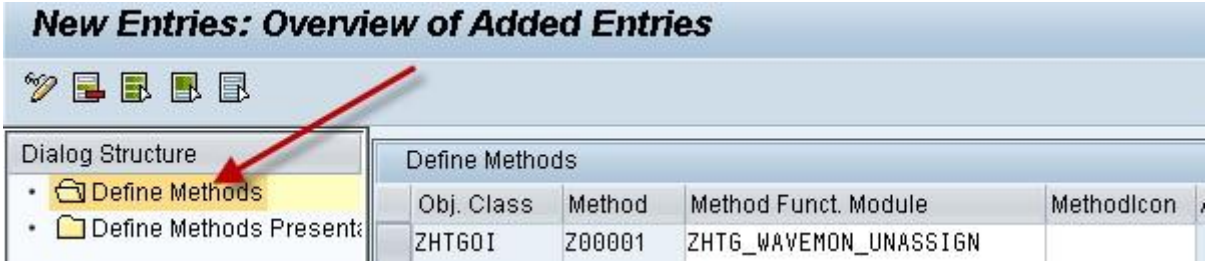


Figure: Define New Object Class Method for Unassigning Waves

Maintain new entries as described in the table below:

Obj. Class	Method	Method Function Module	MethodIcon
ZHTGOI	Z00001	ZHTG_WAVEMON_UNASSIGN	

AutoRefrsh	Text	Presentation Text	Pushb. Text
X	Unassign from Wave XT	Unassign Wav XT	UnassWav

Tip

Use the F4 help of the fields in the column *MethodIcon* for selecting the appropriate icon

The custom object class *ZHTGOI* must contain all object class methods of the object class *WHRITO*:

Method	Method Function Module	Text
M00002	/SCWM/WAVEMON_ASSIGN	Assign to Wave
M00003	/SCWM/WHR_OUT_TREE	Display Item Hierarchy
M00004	/SCWM/WHRITEM_OUT_QUANT_ADJ	Adjust Quantity
M00005	/SCWM/WHRITEM_OUT_QUANT_ADJ_OD	Adjust Quantity and Create OD
M00006	/SCWM/WHRITEM_OUT_BIN_DET	Det. Rough Bin/ Staging Area

Scroll down to the methods that belong to this object class. Mark all required entries and choose the icon *Copy As* (`F6`).

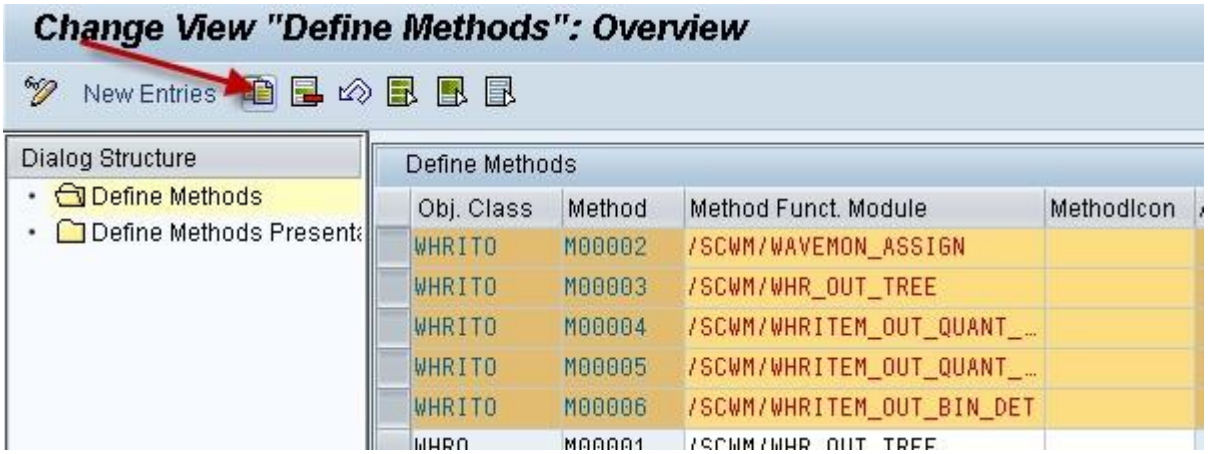


Figure: Copy Object Class Methods from Template to Custom Object Class

Overwrite the value in the column *WHRITO* with *ZHTGOI* and press `ENTER`. Save your changes. Return to the overview screen. Scroll down the method list and check your object class methods.

Define Methods

Obj. Class	Method	Method Funct. Module	MethodIcon	AutoRefrsh	Text
ZHTGOI	M00002	/SCWM/WAVEMON_ASSIGN		☑	Assign to Wave
ZHTGOI	M00003	/SCWM/WHR_OUT_TREE		☐	Display Item Hierarch
ZHTGOI	M00004	/SCWM/WHRITEM_OUT_QUANT_...		☑	Adjust Quantity
ZHTGOI	M00005	/SCWM/WHRITEM_OUT_QUANT_...		☑	Adjust Quantity and C
ZHTGOI	M00006	/SCWM/WHRITEM_OUT_BIN_DET		☑	Det. Rough Bin/ Stag
ZHTGOI	Z00001	ZHTG_WAVEMON_UNASSIGN		☑	Unassign from Wave

Figure: Available Methods for Object Class ZHTGOI

5.1.7 Define Method Presentation of Custom Object Class Method

The new object class method *Z00001* (*Unassign from Wave XT*) shall be added below the method *M00002* (*Assign to Wave*).

Execute the Customizing activity *Define Object Class Methods*. On the following screen double-click the folder *Define Methods Presentation*. Press the button *New Entries*.

Dialog Structure
- Define Methods
- Define Methods Presenta

Define Methods Presentation

Wa...	Monitor	Obj. Class	Sequence	Method	Mthd Pres.
****	ZHTG	ZHTGOI	1	M00002	Accumulat
****	ZHTG	ZHTGOI	2	Z00001	Accumulat
****	ZHTG	ZHTGOI	3	M00003	Accumulat
****	ZHTG	ZHTGOI	4	M00004	Accumulat
****	ZHTG	ZHTGOI	5	M00005	Accumulat
****	ZHTG	ZHTGOI	6	M00006	Accumulat

Figure: Definition of Method Presentations for Object Class ZHTGOI

Maintain the new entries as described in the table below:

Wareh.	Monitor	Obj. Class	Sequence	Method	Mthd Pres.
****	ZHTG	ZHTGOI	1	M00002	Accumulated methods menu button
****	ZHTG	ZHTGOI	2	Z00001	Accumulated methods menu button
****	ZHTG	ZHTGOI	3	M00003	Accumulated methods menu button
****	ZHTG	ZHTGOI	4	M00004	Accumulated methods menu button
****	ZHTG	ZHTGOI	5	M00005	Accumulated methods menu button
****	ZHTG	ZHTGOI	6	M00006	Accumulated methods menu button

The appearance of the methods in the monitor can be influenced using the entries in this table. The sequence of the methods can be maintained by using the *SQNCE* field.

The field *METHOD_PRES* is used to determine how the method is to be displayed. All possible values are domain values (domain */SCWM/D0_METHOD_PRES*).

Fix. Val	Short Descript.
0	Normal pushbutton with icon
1	Normal pushbutton with text
2	Normal pushbutton with icon and text
3	Accumulated methods menu button

The methods that are assigned to the domain fixed value „3" (Accumulated methods menu button) are displayed like this:

Figure: New Object Class Method Assigned to Accumulated Methods Menu Button

5.1.8 Exchange Object Class of Monitor Node against Custom Object Class

Change the custom node profile *ZHTGODOI* that belongs to the monitor node for outbound delivery order items. Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Define Nodes*. On the following screen double-click the folder *Define Node Profile*. Scroll down to the custom node profile *ZHTGODOI*. Exchange the object class *WHRITO* with the custom object class *ZHTGOI*:

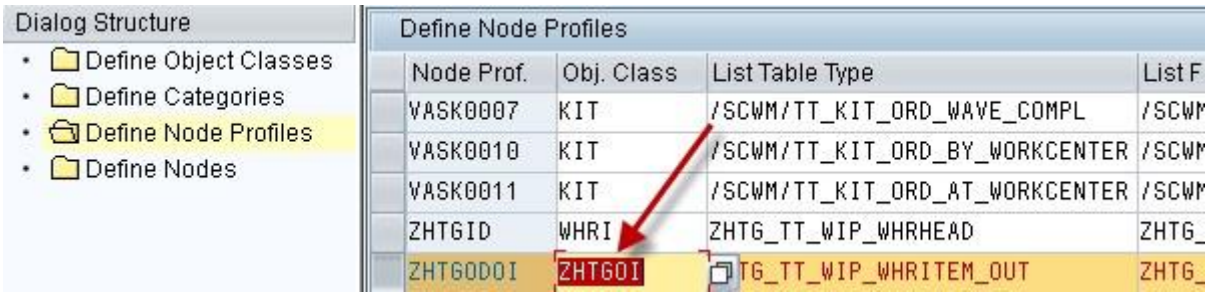


Figure: Assignment of Custom Object Class of Custom Node Profile

5.1.9 Testing

Start the custom warehouse management monitor *ZHTG* in transaction */SCWM/MON*. Open the monitor tree branch *Outbound -> Documents -> Outbound Delivery Order*. Double-click on the custom node *Outbound Delivery Ord. Item XT*. Enter your selection criteria and execute the query. Mark the entries in the result list and call the context menu of the icon *More Methods*. Choose the method *Unassign from Wave XT*.

Figure: Execution of Custom Method for Removal of Wave Assignments

If the marked outbound delivery order items are assigned to waves with the status *Initial* or *Locked*, the system removes these wave assignments. The system will display error messages in the application log popup screen if it is not possible to remove the wave assignments.

6. Add New Application Content

The previous chapters of this document contained basic steps for developing custom functionality for warehouse management monitors. The following chapters provide additional information for creating new content for monitors.

6.1 Customize Own Monitors

The structure and content of warehouse management monitors is maintained in Customizing. This allows great flexibility in designing specific custom monitors without the need to modify the existing SAP standard coding. Starting with release 7.02, SAP has provided a graphical tool for maintaining warehouse management monitor content. Open the Customizing menu path *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor* and execute the Customizing activity *Customize Monitor Tree*.

Figure: Customizing Nodes for Customizing the Warehouse Management Monitor

The graphical maintenance tool allows you to create custom monitors by copying nodes from the SAP standard monitors and creating new category nodes. You can also change the position of nodes in the monitor tree. The node data and node key are accessible by double-clicking the node.

If the content of copied nodes has to be changed, you must execute other activities in the warehouse management monitor Customizing.

6.2 Example: Create Custom Monitor for New Custom Nodes

The new custom monitor *ZHTG_S* shall contain custom nodes for outbound delivery order items and inbound deliveries. Both custom nodes shall be grouped in a new category *ZHTG_C*. The inbound

delivery node shall appear in the first position, and the outbound delivery node in the second. The custom monitor shall be available for all warehouses and shall not inherit the SAP standard monitor.

Figure: Planned Node Hierarchy of Simple Custom Monitor ZHTG_S

6.2.1 Required Steps

The following steps are required to create this simple custom monitor:

- Create a new monitor *ZHTG_S*
- Create a new category *ZHTG_C*
- Define new object classes
- Define new object class methods
  - Reuse the object class method for unassigning outbound delivery order items from waves (chapter 5.1)
- Define an object class method presentation for the new monitor
- Create new node profiles
  - Reuse the node profiles *ZHTGID* (chapter 4.3.5) and *ZHTGODOI* (chapter 4.2.5)
- Define monitor nodes
  - Reuse the custom nodes for outbound delivery order items (chapter 4.2) and inbound deliveries (chapter 4.3)
- Create a node hierarchy

6.2.2 Define Custom Monitor ZHTG_S

The custom monitor can be created using either the Customizing activity *Customize Monitor* or the Customizing activity *Define Monitors*. The latter Customizing activity allows you to maintain additional information and is used in our example.

Execute the Customizing activity *Define Monitors*. Mark the folder *Define Monitor* and press the button *New Entries*.

Figure: Creation of Custom Monitor ZHTG_S

Maintain new entries as described in the table below:

Warehouse Number	Monitor	StdMonitor	+	Text
****	ZHTG_S			How To Guide: Simple Monitor

The following fields can be maintained:

Column Name	Description

Wareh.	Number of warehouse or warehouse complex for which this monitor variant shall be used. If monitor variant shall be useable for all warehouses, enter placeholder ****
Monitor	Name of your monitor variant
StdMonitor	Name of standard monitor that is merged with monitor
+	If checkbox is ticked, monitor tree uses short texts. Otherwise long texts are used and displayed in monitor tree
Text	Description for your monitor variant

Recommendation

Creating node hierarchies is easy when you use the Customizing activity *Customize Monitor* instead of the view maintenance option *Define Node Hierarchy*. So you should use the graphical maintenance application instead of the view maintenance option.

6.2.2.1 Involved Customizing Tables and Maintenance Views

The tree hierarchy data is stored in the tables */SCWM/TMONITOR* and */SCWM/TMONITORT*:

- */SCWM/TMONITOR* - monitors
- */SCWM/TMONITORT* - monitor descriptions Corresponding maintenance view:
- */SCWM/V_MONITOR* - warehouse management monitors

Relevant field values of Customizing tables */SCWM/TMONITOR* and */SCWM/TMONITORT*:

Table Name	Field	Data Element	Description
/SCWM/TMONITOR	LGNUM	/SCWM/LGNUM	Warehouse Number/Warehouse Complex
/SCWM/TMONITOR	MONITOR	/SCWM/DE_MONITOR	Monitor
/SCWM/TMONITOR	STAND_MONITOR	/SCWM/DE_STAND_MONITOR	Standard Monitor
/SCWM/TMONITOR	SHORT_TEXT	/SCWM/DE_MONTREE_SHORT_TEXT	Display short text in monitor tree
/SCWM/TMONITORT	TEXT	/SCWM/DE_TEXT	Text

6.2.3 Create New Category ZHTG_C

Categories are used to group nodes and node branches. Category folders are displayed at the first level of the navigation tree hierarchy. They can contain various child nodes. Execute the Customizing activity *Define Nodes* and double-click the folder *Define Categories* in the navigation tree.

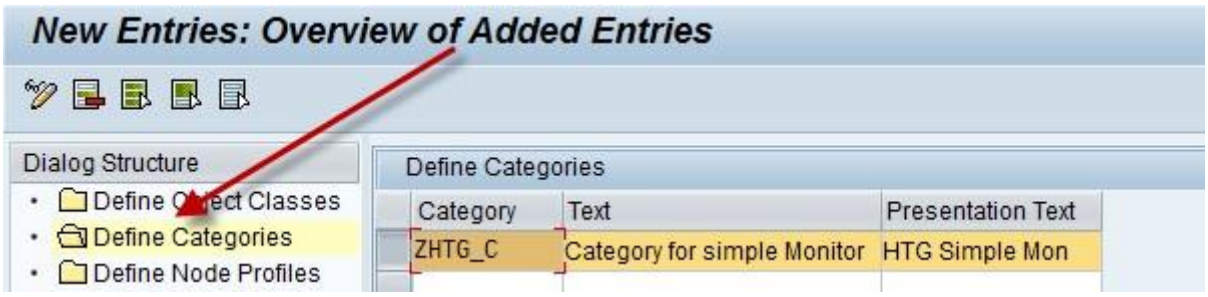


Figure: Definition of new Custom Category ZHTG_C

Maintain a new entry as in the following table:

Category	Text	Presentation Text
ZHTG_C	Category for simple Monitor	HTG Simple Mon

6.2.3.1 Involved Customizing Tables and Maintenance Views

Category data is stored in two tables:

- */SCWM/TCATEGORY* - Categories
- */SCWM/TCATEGORYT* - Category Descriptions Corresponding maintenance view:
- */SCWM/V_CATEGORY* - Categories

Relevant field values of the Customizing tables */SCWM/TCATEGORY* and */SCWM/CATEGORYT*:

Table Name	Field	Data Element	Description
/SCWM/CATEGORY	CATEGORY	/SCWM/DE_CATEGORY	Category
/SCWM/CATEGORYT	TEXT	/SCWM/DE_TEXT	Text
/SCWM/CATEGORYT	PTEXT	/SCWM/DE_PTEXT	Presentation text

6.2.4 Define New Object Class ZHTGID

Whereas the custom object class ZHTGOI is reused for outbound delivery items (chapter 5.1.5), the custom object class ZHTGID is created for inbound deliveries.

Go to the Customizing activity *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor* → *Define Nodes* to create the corresponding Customizing entries.

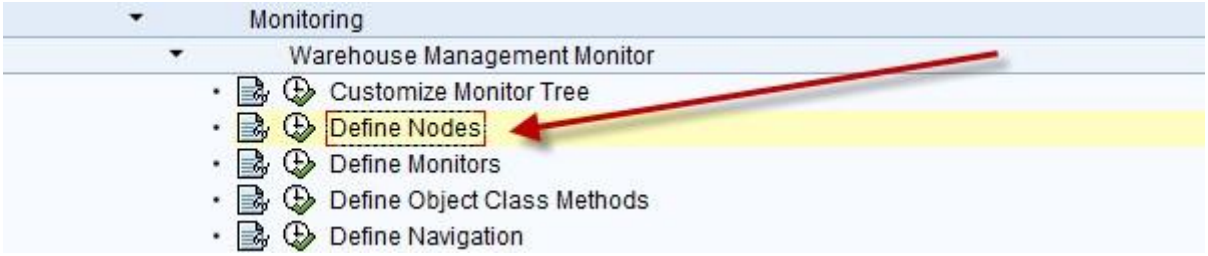


Figure: Customizing Activity for Creating Monitor Nodes

Execute this Customizing activity. Mark the folder *Define Object Classes* and choose *New Entries*. Maintain the entries as described in the following table and save your changes:

Obj. Class	Text	Presentation Text
ZHTGID	Inbound Deliveries (ObjCl)	InbDlvObjCl

6.2.4.1 Involved Customizing Tables and Maintenance Views

Object class data is stored in two tables:

- */SCWM/TOBCLS* - Object Classes
- */SCWM/TOBCLST* - Object Class Descriptions Corresponding maintenance view:

- */SCWM/V_OBCLS* - Object Classes

Relevant field values of Customizing tables */SCWM/TOBCLS* and */SCWM/TOBCLST*:

Table Name	Field	Data Element	Description
/SCWM/TOBCLS	OBCLS	/SCWM/DE_OBCLS	Object Class
/SCWM/TOBCLST	TEXT	/SCWM/DE_TEXT	Text
/SCWM/TOBCLST	PTEXT	/SCWM/DE_PTEXT	Presentation Text

6.2.5 Define New Object Class Methods

The methods of the object class ZHTGOI are reused (chapter 5.1.6). The methods to display inspection documents for inbound deliveries for the object class ZHTGID are created as follows: In Customizing, go to *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*.

Figure: Customizing Activity for Creating Object Class Methods

Execute the Customizing Activity *Define Object Class Methods*. On the following screen double-click the folder *Define Methods*. Choose *New Entries*. Maintain new entries as described in the following table:

Obj. Class	Method	Method Function Module	MethodIcon
ZHTGID	ZIDM01	/SCWM/INSP_MON	

AutoRefrsh	Text	Presentation Text	Pushb. Text
	Display Inspection Document	Displ. Insp.Doc	Insp.Doc

6.2.6 Define Methods Presentation

Execute the Customizing activity *Define Object Class Methods*. On the following screen double-click the folder *Define Methods Presentation*. Choose *New Entries*. Maintain new entries as described in the following table:

Wareh.	Monitor	Obj. Class	Sequence	Method	Mthd Pres.
****	ZHTG_S	ZHTGID	1	ZIDM01	Accumulated methods menu button
****	ZHTG_S	ZHTGOI	1	M00002	Accumulated methods menu button
****	ZHTG_S	ZHTGOI	2	Z00001	Accumulated methods menu button
****	ZHTG_S	ZHTGOI	3	M00003	Accumulated methods menu button
****	ZHTG_S	ZHTGOI	4	M00004	Accumulated methods menu button
****	ZHTG_S	ZHTGOI	5	M00005	Accumulated methods menu button

****	ZHTG_S	ZHTGOI	6	M00006	Accumulated methods menu button

The appearance of the methods in the monitor can be influenced using the entries in this table. The sequence of the methods can be maintained using the *SQNCE* field.

The field *METHOD_PRES* is used to determine how the method is to be displayed. All possible values are domain values (domain */SCWM/D0_METHOD_PRES*):

Fix. Val	Short Descript.
0	Normal pushbutton with icon
1	Normal pushbutton with text
2	Normal pushbutton with icon and text
3	Accumulated methods menu button

Methods that are assigned to the domain fixed value „3" (Accumulated methods menu button) are displayed like this:

Figure: New Object Class Method Assigned to Accumulated Methods Menu Button

6.2.6.1 Involved Customizing Tables and Maintenance Views

Object class method data is stored in the following table:

- */SCWM/TOBCLS_MTH* - Object Class Methods Presentation Corresponding maintenance view:
- */SCWM/V_OBCLS_MT* - Methods Presentation

Relevant field values of the Customizing table */SCWM/TOBCLS_MTH*:

Field	Data Element	Description
LGNUM	/SCWM/LGNUM	Warehouse Number/Warehouse Complex
MONITOR	/SCWM/DE_MONITOR	Monitor
OBCLS	/SCWM/DE_OBCLS	Object Class
SQNCE	/SCWM/DE_SQNCE	Sequence
METHOD	/SCWM/DE_METHOD	Object Class Method for Monitor
METHOD_PRES	/SCWM/DE_METHOD_PRES	Method Presentation
LGNUM	/SCWM/LGNUM	Warehouse Number/Warehouse Complex
MONITOR	/SCWM/DE_MONITOR	Monitor

6.2.7 Define Node Profiles for Monitor Tree Nodes

Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Define Nodes*. On the following screen double-click the folder *Define Node Profile*. Choose *New Entries*. Maintain a new entry according to the following tables and save your changes:

Node Prof.	Obj. Class	List Table Type	List Funct. Module
ZHTGID	ZHTGID	ZHTG_TT_WIP_WHRHEAD	ZHTG_WHRHEAD_MON_XT

Form Funct. Module	Form Structure	Dynpro Program
	ZHTG_S_WIP_WHRHEAD	SAPLZHTG

Dynpro No.	Text	Presentation Text	Disable
0110	Inbound Delivery XT	Inb. Deliv. XT	

6.2.7.1 Involved Customizing Tables and Maintenance Views

Node profile data is stored in two tables:

- */SCWM/TNODE_PRF* - Node Profiles
- */SCWM/TNODE_PRFT* - Node Profile Descriptions Corresponding maintenance view:
- */SCWM/V_NODE_PRF* Node Profiles

Relevant field values of Customizing tables */SCWM/TNODE_PRF* and */SCWM/TNODE_PRFT*:

Table Name	Field	Data Element	Description
/SCWM/TNODE_PRF	NODE_PRF	/SCWM/DE_NODE_PRF	ID of node profile
/SCWM/TNODE_PRF	OBCLS	/SCWM/DE_OBCLS	Object class of the node profile
/SCWM/TNODE_PRF	LIST_FM	/SCWM/DE_LIST_FM	Name of the function module that gets the data for the ALV List view
Table Name	**Field**	**Data Element**	**Description**
/SCWM/TNODE_PRF	LIST_TT	/SCWM/DE_LIST_TT	Table type of the exporting parameter ET_DATA of the LIST_FM
/SCWM/TNODE_PRF	FORM_FM	/SCWM/DE_FORM_FM	Name of the function module that gets the data for the form view
/SCWM/TNODE_PRF	FORM_ST	/SCWM/DE_FORM_ST	Structure type of the exporting parameter ET_DATA of the LIST_FM
/SCWM/TNODE_PRF	DYNP_PR	/SCWM/DE_DYNP_PR	Name of the main program that contains the dynpro for the form view
/SCWM/TNODE_PRF	DYNP_NR	/SCWM/DE_DYNP_NR	Number of the dynpro that is to be used for the form view

/SCWM/TNODE_PRF	DISABLE	/SCWM/DE_DISABLE	Node will not be directly accessible. Instead, it will only be available for drill down.
/SCWM/TNODE_PRFT	TEXT	/SCWM/DE_TEXT	Text
/SCWM/TNODE_PRFT	PTEXT	/SCWM/DE_PTEXT	Presentation Text

Important

If the fields *FORM_FM* and *FORM_ST* are not filled, it will be assumed that the form view uses the same structure type as ALV list view.

6.2.8 Define Monitor Tree Nodes

After node profiles are created it is now possible to create the new nodes and assign profiles to them. Double-click the folder *Define Nodes*. Choose *New Entries*.

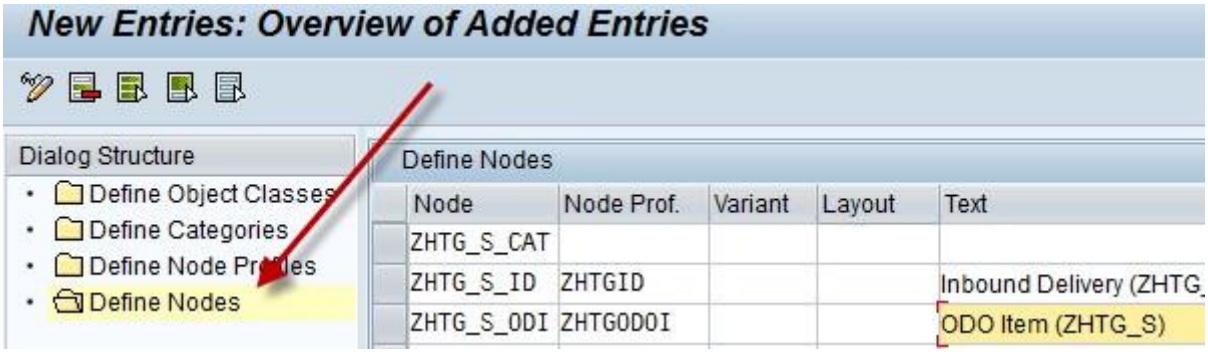


Figure: Create Custom Nodes

Maintain new entries according to the following table and save your changes:

Node	Category	Node Prof.	Variant	Layout	Text
ZHTG_S_CAT	ZHTG_C				
ZHTG_S_ID		ZHTGID			Inbound Delivery (ZHTG_S)
Node	**Category**	**Node Prof.**	**Variant**	**Layout**	**Text**
ZHTG_S_ODI		ZHTGODOI			ODO Item (ZHTG_S)

6.2.8.1 Involved Customizing Tables and Maintenance Views

Node data is stored in two tables:

- */SCWM/TNODE* - Nodes
- */SCWM/TNODET* - Node Descriptions Corresponding maintenance view:
- */SCWM/V_NODE* - Nodes

Relevant field values of Customizing tables */SCWM/TNODE* and */SCWM/TNODET*:

Table Name	Field	Data Element	Description
/SCWM/TNODE	NODE	/SCWM/DE_NODE	Node

/SCWM/TNODE	CATEGORY	/SCWM/DE_CATEGORY	Category
/SCWM/TNODE	NODE_PRF	/SCWM/DE_NODE_PRF	Node Profile
/SCWM/TNODE	VARID	VARIANT	Report variant
/SCWM/TNODE	LAYOUT	SLIS_VARI	ALV Layout variant
/SCWM/TNODET	TEXT	/SCWM/DE_TEXT	Text
/SCWM/TNODET	PTEXT	/SCWM/DE_PTEXT	Presentation Text

6.2.9 Define Node Hierarchy for Monitor ZHTG_S

Using the Customizing activity *Customize Monitor Tree* is the easiest way to define new node hierarchies. Open the Customizing node *Extended Warehouse Management* → *Monitoring* → *Warehouse Management Monitor*. Execute the Customizing activity *Customize Monitor Tree*. In the right column choose *Open Monitor* to load the custom monitor ZHTG_S. Choose the menu entry *Node Pool -> Display All Nodes*:

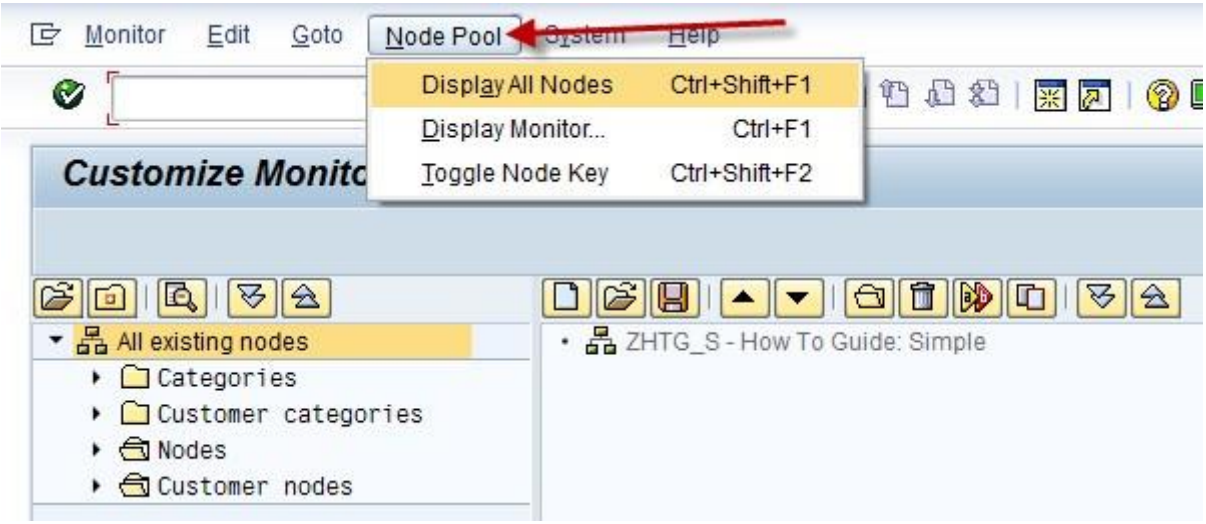


Figure: Loading Custom Nodes for Building Monitor Tree Hierarchy

Open the folder *Customer Categories*. Drag and drop the node ZHTG_S_CAT to the monitor ZHTG_S:

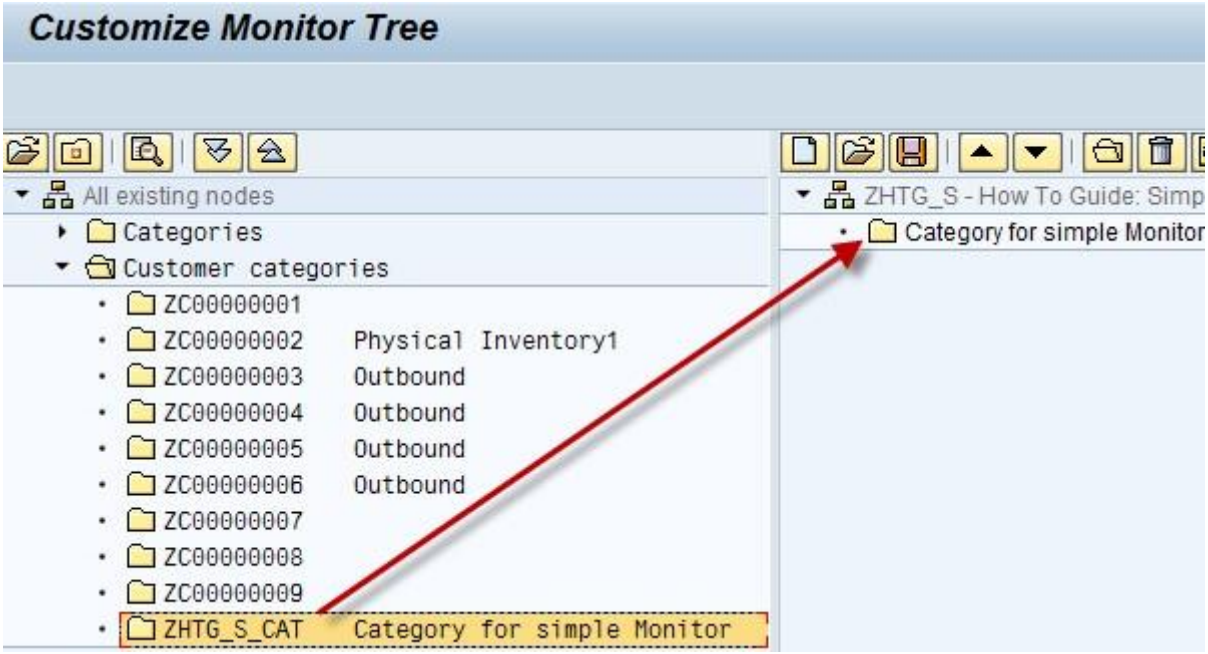


Figure: Creation of Category Node by Drag & Drop

Then open the folder *Customer Nodes*. Drag and drop the nodes ZHTG_S_ID and ZHTG_S_ODI into the category folder of the custom monitor ZHTG_S:

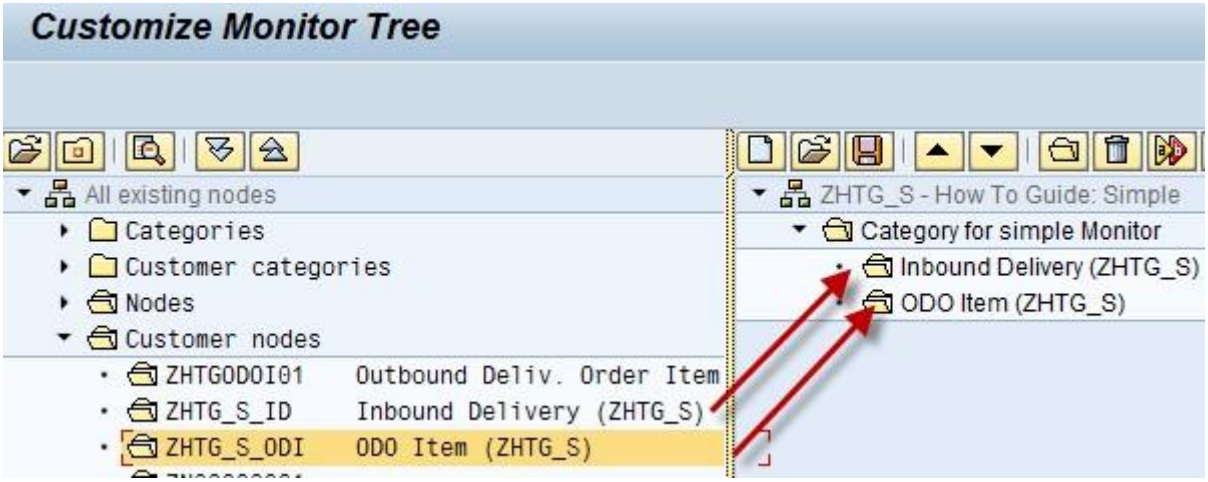


Figure: Creation of Nodes for Inbound Deliveries and Outbound Delivery Order Items Drag & Drop

Save your changes.

<u>Notes</u>:

- You can change node positions by selecting the node and pressing the arrow key buttons (“*Move Node Up*”, “*Move Node Down*”).
- Functions for renaming, duplicating, or deleting nodes are also assigned to pushbuttons.

6.2.9.1 Creation of Node Hierarchy by View Maintenance

The Customizing activity *Define Monitor* allows creation of node hierarchies as well. Mark the monitor ZHTG_S and double-click on the folder *Define Node Hierarchy*. Choose *New Entries* and maintain new entries according to the following table:

HigherNode	Lower Node	Sequence	Hide Node	Hide PB
ROOT	ZHTG_S_CAT	1		
ZHTG_S_CAT	ZHTG_S_ID	1		
ZHTG_S_CAT	ZHTG_S_ODI	2		

Important

Category nodes on the top level (like *Outbound* in this example) have to be children of the *ROOT* node.

The hierarchy of the tree is derived from the relationship between parent and child. If a parent node has more than one child, the sequence of the child node entries is determined by the *SQNCE* field.

6.2.9.2 Involved Customizing Tables and Maintenance Views

Tree hierarchy data is stored in one table:

- */SCWM/TTREE* Warehouse Management Monitor Hierarchy Tree Corresponding maintenance view:
- */SCWM/V_TREE* Warehouse Management Monitor Node Hierarchies

Relevant field values of Customizing table */SCWM/TTREE*:

Field	Data Element	Description
LGNUM	/SCWM/LGNUM	Warehouse Number/Warehouse Complex
MONITOR	/SCWM/DE_MONITOR	Name of monitor
PNODE	/SCWM/DE_PNODE	Node ID of parent node (Higher-Level Node)
CNODE	/SCWM/DE_CNODE	Node ID of child node (Lower-Level Node)
SQNCE	/SCWM/DE_SQNCE	Sequence of entry
FLG_PB	/SCWM/DE_FLG_PB	Present child nodes with pushbuttons in the ALV toolbar (for Lower-Level Node in List View Toolbar)
FLG_HIDE	/SCWM/DE_FLG_HIDE	Hide Node

6.2.10 Testing New Custom Monitor ZHTG_S

Start the warehouse management monitor (transaction /SCWM/MON) and choose monitor ZHTG_S. Execute queries for inbound deliveries and outbound delivery order items:

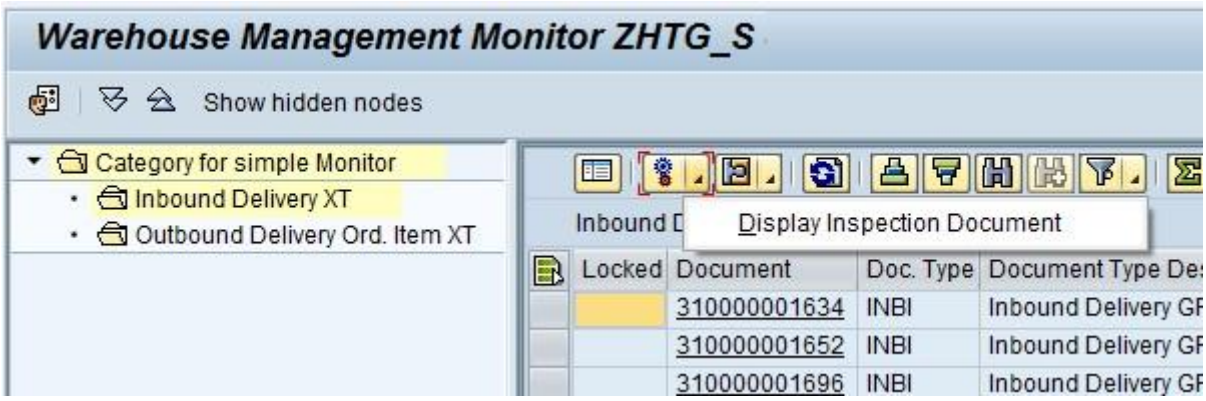


Figure: Query for Inbound Deliveries using New Custom Monitor ZHTG_S

6.3 Function Modules for New Application Content

The function modules of an application provider have to fulfill several guidelines. These guidelines comprise conventions for the interface of the function modules used and a few mandatory processing steps that have to be performed by every function module to be used within the warehouse management monitor.

Function modules belonging to one branch of the tree can be grouped together in one function group. The type-pool *RSDS* must be used in the top include of the function group.

The function group */SCWM/WIP_WAVE* exemplifies the wave branch of the outbound view.

6.3.1 Selection Screens

The selection screens for the function modules have to be created in the top include of the function group that the function modules belong to. The standard means such as *PARAMETERS* and *SELECT-OPTIONS* must be used there. If you are establishing a hierarchy of nodes, you should use frames to distinguish the selection criteria of the different nodes.

A new function key that erases the content of all input fields should be added to every selection screen. For this purpose, a code snippet has to be added to the top include of the function group. This uses the events *LOAD-OF-PROGRAM* and *AT SELECTION-SCREEN*. In addition, the function key has to be added to each defined selection screen. This can be seen in the example below.

6.3.1.1 Example Selection Screen

An example of a selection screen might look like the code snippet below. The selection screen provides selection criteria for wave items. Since the selection screen is for a child of the node wave, the selection criteria of the wave node must be included. This can be achieved by using the statement *SELECTION-SCREEN INCLUDE xxx*. This means that *SELECT-OPTIONS* and *PARAMETERS*, or even whole blocks that have already been defined can be reused.

Source Coding of Sample Selection Screen

```
* Dynpro 200 - Wave item selection
DATA: wave_itm_sel TYPE /scwm/de_wave_itm,
docno_sel     TYPE /scwm/de_docno_r,
itmno_sel     TYPE /scwm/de_itemno_r,
type_2st      TYPE /scwm/de_typ2st,
stat_cr_sel   TYPE /scwm/de_statcr.

SELECTION-SCREEN BEGIN OF SCREEN 200 AS WINDOW.
SELECTION-SCREEN INCLUDE BLOCKS frm101.
SELECTION-SCREEN BEGIN OF BLOCK frm202 WITH FRAME TITLE text-002.
SELECT-OPTIONS:
  s_waitm    FOR wave_itm_sel,
s_docno    FOR docno_sel,
```

Source Coding of Sample Selection Screen

```
  s_itemno   FOR itmno_sel,
s_ty2st    FOR type_2st,   s_crst
FOR stat_cr_sel. SELECTION-SCREEN
END OF BLOCK frm202.
SELECTION-SCREEN INCLUDE PARAMETERS:
p_avdsel.
SELECTION-SCREEN: FUNCTION KEY 1. SELECTION-
SCREEN END   OF SCREEN 200.

LOAD-OF-PROGRAM * erase button
functxt-icon_id   = icon_erase.
  SELECT SINGLE quickinfo FROM icont
                          INTO functxt-quickinfo
                          WHERE langu = sy-langu
                            AND id    = functxt-icon_id.   sscrfields-
functxt_01 = functxt.

AT SELECTION-SCREEN.
* react on erase button
CASE sscrfields-ucomm.
    WHEN 'FC01'.
      CALL FUNCTION
'/SCWM/DYNPRO_ELEMENTS_CLEAR'         EXPORTING
iv_repid = sy-repid.
  ENDCASE.
```

6.3.2 Interface

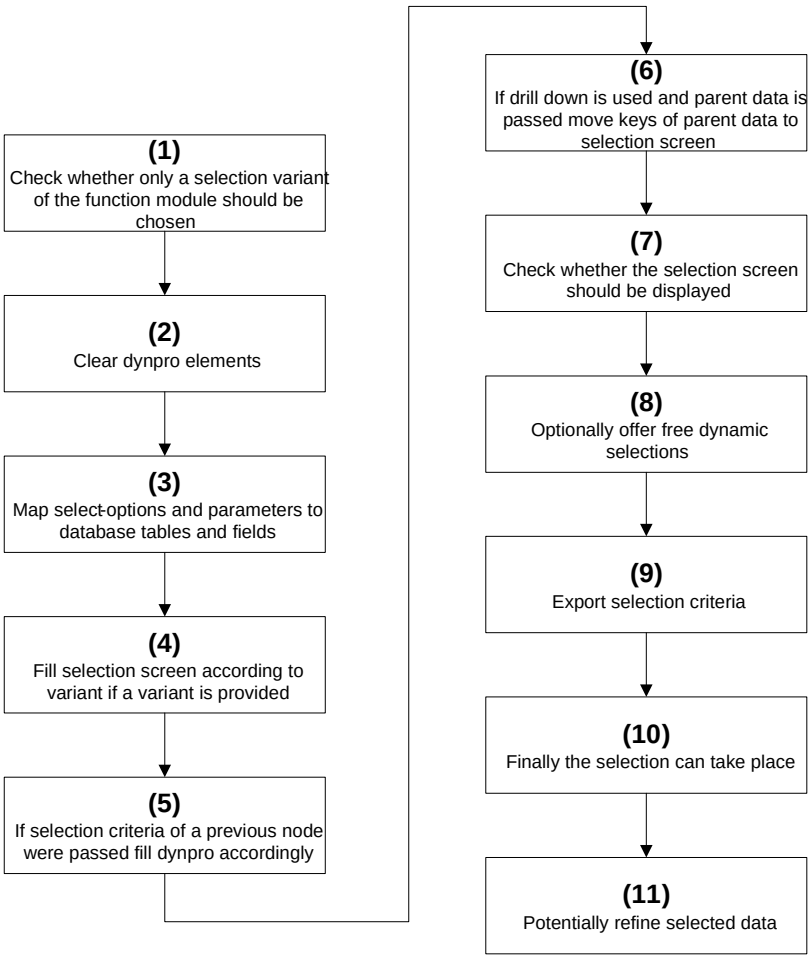
Each function module has to provide an interface with the following parameters:

Parameter	Type	Associated Type	Description
IV_LGNUM	Imp.	/SCWM/LGNUM	Warehouse number/complex
IV_VARIANT (optional)	Imp.	VARIANT	Variant the function module should be called with
IV_MODE	Imp.	/SCWM/DE_MON_FM_MODE	Mode for function module execution: 1 = normal; 2 = no selection screen; 3 = only choose selection variant; 4 = refresh
IT_DATA_PARENT (optional)	Imp.	Table type of selected data of parent node	Table consisting of (parent) entries selected for drill down
EV_RETURNCODE	Exp.	XFELD	Should be set to „X" if the user cancels the selection
EV_VARIANT	Exp.	VARIANT	Variant that has been selected
ET_DATA	Exp.	Table type of selected data of current node	Table consisting of entries that comply with the specified selection criteria
CT_TAB_RANGE *(optional)*	Chang.	RSDS_TRANGE	Table consisting of selection options of previous nodes

Parameter	Type	Associated Type	Description
CT_FIELDCAT *(optional)*	Chang.	LVC_T_FCAT	ALV field catalog

6.3.3 Mandatory Processing Steps

The following steps must be implemented in each function module:



An example is provided for each step:

1. Firstly, a check must be carried out to see if only a selection variant of the function module is to be chosen. If the importing parameter *IV_MODE* is set to „3," the function module *RS_VARIANT_CATALOG* must be called. After the call, the function module should be quit.

```
IF iv_mode = 3.
  CALL FUNCTION 'RS_VARIANT_CATALOG'
EXPORTING
      report                      = lv_repid
*     NEW_TITLE                   = ' '
      DYNNR                       = lc_dynnr
*     POP_UP                      = ' '
    IMPORTING
      SEL_VARIANT                 = ev_variant
    EXCEPTIONS
      NO_REPORT                   = 1
      REPORT_NOT_EXISTENT         = 2
      REPORT_NOT_SUPPLIED         = 3
      NO_VARIANTS                 = 4
      NO_VARIANT_SELECTED         = 5
```

```
      VARIANT_NOT_EXISTENT       = 6
OTHERS                     = 99.
  IF sy-subrc <> 0.
    MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
  ENDIF.
  RETURN.
ENDIF.
```

2. The data of the selection screen has to be initialized using the function module */SCWM/DYNPRO_ELEMENTS_CLEAR*.

```
CALL FUNCTION '/SCWM/DYNPRO_ELEMENTS_CLEAR'
EXPORTING       iv_repid = lv_repid.
```

3. The selection options and parameters of the selection screen have to be mapped to fields of the database table that contains the data. You can freely name the selection options and the parameters of the selection screen. To be able to fill the selection criteria while navigating through a branch of the tree, the monitor has to be informed how the selection options and parameters are linked to the fields of the database tables. For this purpose, a local table of type */SCWM/TT_MAP_SELOPT2FIELD* must be filled. For later processing, the field *IS_KEY* has to be set for key fields of the described table. In addition, the field *IS_TIMESTAMP* can be used to convert a timestamp interval into time and date from and time and date to. By doing so, the function module */SCWM/RANGETAB2SELOPT* is able to convert timestamp select-options that have been passed by a parent node, into the corresponding parameters for dates and times on the selection screen.

```
MOVE: '/SCWM/WAVEHDR'   TO ls_mapping-tablename,   “database table name
      'WAVE'          TO ls_mapping-fieldname,   “database field name
      'S_WAVE'        TO ls_mapping-selname,     “select options on dynpro      „X“          TO ls_mapping-
is_key.     “field is key field
 APPEND ls_mapping      TO lt_mapping.
 CLEAR  ls_mapping.
 MOVE: '/SCWM/WAVEHDR'   TO ls_mapping-tablename,   “database table name
      'CUTOFF_DT'      TO ls_mapping-fieldname, . “database field name
      'S_WACDT'        TO ls_mapping-selname,     “select options on dynpro
      'X'            TO ls_mapping-is_timestamp,”field is timestamp
      'P_CDTFR'        TO ls_mapping-p_date_from, “parameter for date from
      'P_CTMFR'        TO ls_mapping-p_time_from, “parameter for time from
      'P_CDTTO'        TO ls_mapping-p_date_to,   “parameter for date to
      'P_CTMTO'        TO ls_mapping-p_time_to.   “parameter for time to
 APPEND ls_mapping      TO lt_mapping.
```

This may be swapped for a form routine.

4. If the parameter *IV_VARIANT* is provided, the function module *RSDS_SUPPORT_ SELECTIONS* has to be called to fill the selection parameters accordingly.

```
   CALL FUNCTION 'RS_SUPPORT_SELECTIONS'
EXPORTING
      report              = lv_repid         variant            =
iv_variant      EXCEPTIONS
      variant_not_existent = 1        variant_obsolete
= 2        OTHERS              = 3.
```

5. Thereafter, you must check whether the parameter *CT_TAB_RANGE* is empty. If it is, the function module */SCWM/RANGETAB2SELOPT* has to be called. This function module fills the selection criteria of the selection screen according to the parameter *CT_TAB_RANGE* which has been passed by previous nodes.

```
   CALL FUNCTION '/SCWM/RANGETAB2SELOPT'
EXPORTING
```

```
      iv_repid     = lv_repid        iv_lgnum     =
iv_lgnum        it_mapping   = lt_mapping
CHANGING        ct_tab_range = ct_tab_range.
```

6. If the parameter *IT_DATA_PARENT* is not empty, the corresponding selection options of key fields of parent data have to be filled in accordance with table *IT_DATA_PARENT*. The monitor framework fills this importing parameter if the user uses one of the pushbuttons to drill down. The function module */SCWM/FILL_SELOPT_BY_KEYS* fills the selection options based on the key fields specified in the table *LT_MAPPING*.

```
   CALL FUNCTION '/SCWM/FILL_SELOPT_BY_KEYS'      EXPORTING
      iv_repid            = lv_repid        it_mapping          = lt_mapping
it_data_parent       = it_data_parent.
```

7. Thereafter, you must evaluate whether the selection screen is to be displayed by using the parameter *IV_MODE*. The selection will only have to be displayed if it is set to *1*. If the user pressed *Cancel* on the selection screen, the exporting parameter *EV_RETURNCODE* has to be set to *X* and the function module has to be quit.

```
   IF iv_mode = „1".
                CALL SELECTION-SCREEN lc_dynnr STARTING AT 10 10
                                                    ENDING AT 130 30.
     IF sy-subrc IS NOT INITIAL.
       MOVE 'X' TO ev_returncode.
       RETURN.
     ENDIF.     END
IF.
```

If the application developer wants to provide free dynamic selections, the function module */SCWM/GET_FREE_SELECTIONS* should be used. It should be called via a parameter on the selection screen. This function module requires a table consisting of table names for the free dynamic selections as import parameters. If aliases are used in the select statement, the aliases will also have to be passed. The result of the function module is a table that contains constraints that can be used in the select statement.

```
 IF p_avdsel IS NOT INITIAL.
   MOVE: '/SCWM/WAVEHDR' TO ls_tabname-tabname,
         'WH'            TO ls_tabname-sel_alias.
   APPEND ls_tabname TO lt_tabname.
   MOVE: '/SCWM/WAVEITM' TO ls_tabname-tabname,
         'WI'            TO ls_tabname-sel_alias.
   APPEND ls_tabname TO lt_tabname.
   CALL FUNCTION '/SCWM/GET_FREE_SELECTIONS'
EXPORTING
      it_tabname       =  lt_tabname          IMPORTING
et_whereclause = lt_whereclause.   ENDIF.
```

8. Then the function module */SCWM/SELOPT2RANGETAB* will fill the exporting parameter *CT_TAB_RANGE* in accordance with the specified selection criteria.

```
 CALL FUNCTION '/SCWM/SELOPT2RANGETAB'
EXPORTING
     iv_repid     = lv_repid
     it_mapping   = lt_mapping     IMPORTING
et_tab_range = ct_tab_range.
```

9. Finally, the selection can take place.

10. The selected data may have to be refined to be displayed (e.g. timestamps can be converted to date and time).

6.3.3.1 Coding Example: Function Module ZHTG_WHRHEAD_MON_XT

The following sections of the function module *ZHTG_WHRHEAD_MON_XT* demonstrate how implementation steps can be realized. The complete source code of the function module is listed in the appendix.

Step	Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT

1.	`**********`
	`*  STEP 1: Check whether only selection variant of function module`
	`*`
	`*  should be chosen`
	`*`
	`**********`
	`  IF iv_mode = lc_mode_sel_var.`
	`  CALL FUNCTION 'RS_VARIANT_CATALOG'`
	`EXPORTING`
	`  report  = sy-repid`
	`*  NEW_TITLE  = ' '`
	`  dynnr  = lc_dynnr_0100`
	`*  POP_UP  = ' '  IMPORTING`
	`  sel_variant  = ev_variant`
	`EXCEPTIONS`
	`  no_report  = 1`
	`report_not_existent  = 2`
	`report_not_supplied  = 3`
	`no_variants  = 4`
	`no_variant_selected  = 5`
	`variant_not_existent = 6`
	`OTHERS  = 7.`
	
	`  IF sy-subrc <> 0.`
	`  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno`
	`  WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.`
	`  ENDIF.`
	`  RETURN.`
	`  ENDIF.`
	`**********`
	`....`

2.	`**********`
	`* STEP 2: Clear dynpro screen elements  *`
	`**********`
	`lv_repid = sy-repid.`
	
	`  CALL FUNCTION`
	`'/SCWM/DYNPRO_ELEMENTS_CLEAR'  EXPORTING`
	`iv_repid = lv_repid.`
	`**********`

3.	`**********`

Step	Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
*     STEP 3: Map select-options and parameters to database tables and
*
*     fields
*
**********************************************************************
go_hlp_id->map_whrhead_to_sel_opt(
    CHANGING ct_mapping = lt_mapping ).
   p_lgnum =
iv_lgnum.

**********************************************************************
```

4.

```
**********************************************************************
*   STEP 4: If variant is provided, fill selection screen in accordance
    *
*   with variant
    *
**********************************************************************
  IF NOT iv_variant IS INITIAL.
*   Use the selection criteria from a pre-defined variant without
*   presenting a selection screen
    CALL FUNCTION 'RS_SUPPORT_SELECTIONS'
EXPORTING
        report                  = sy-repid
variant                      =  iv_variant
EXCEPTIONS
        variant_not_existent = 1
variant_obsolete      = 2
OTHERS                = 3.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.     ENDIF.
  ENDIF.
**********************************************************************
```

5.

```
**********************************************************************
*     STEP 5: In case selection criteria of previous node are passed fill
*
*     dynpro with them
*
**********************************************************************
  IF ct_tab_range IS NOT INITIAL.
*     the table it_tab_range contains the selection criteria, which
*     have been passed to the function module
*     these selection criteria should be visible in the selection screen
CALL FUNCTION '/SCWM/RANGETAB2SELOPT'       EXPORTING
        iv_repid      = sy-repid
iv_lgnum      = iv_lgnum
it_mapping    = lt_mapping
CHANGING          ct_tab_range =
ct_tab_range.
  ENDIF.
**********************************************************************
```

6.	```***
	* STEP 6: If drill-down is used and parent data is passed move keys of *
	* parent data to selection screen -> not used in this function module *
	***
	* CALL FUNCTION '/SCWM/FILL_SELOPT_BY_KEYS'
	* EXPORTING
	* iv_repid = lv_repid
	* iv_lgnum = iv_lgnum
	* it_mapping = lt_mapping
	* it_data_parent = it_data_parent.
	
	``

Step	Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT

	``***
	
	``

7.	```***
	* STEP 7: Check whether selection screen should be displayed *
	***
	  IF iv_mode = lc_mode_std_sel.
	* Show selection screen and use the selection criteria entered on
	* the screen. This screen can also be used for definition of a
	* variant (standard functionality of selection-screens)
	  CALL SELECTION-SCREEN lc_dynnr_0100 STARTING AT 10 10
	ENDING AT 130 30.
	  IF sy-subrc IS NOT INITIAL.
	ev_returncode = abap_true.
	  RETURN.
	  ENDIF.
	  ENDIF.
	***
	....```

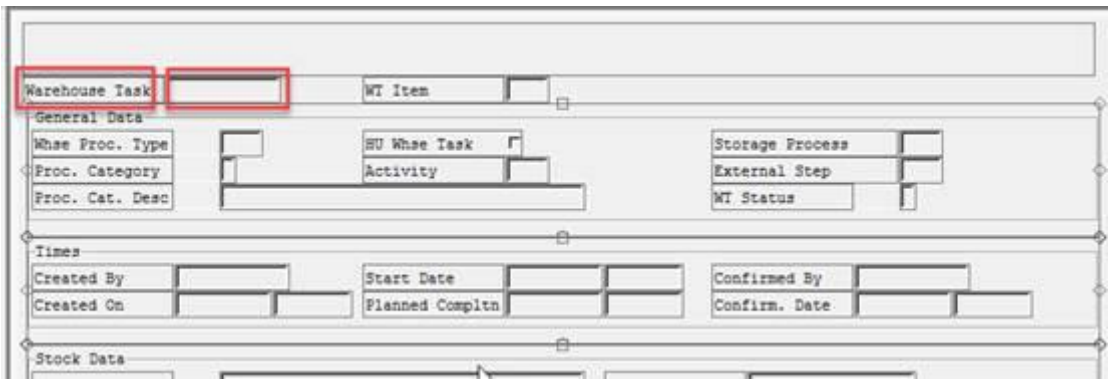
8.	```*** * STEP 8: Optionally offer free dynamic selections -> not used in this * * function module * *** * CALL FUNCTION '/SCWM/GET_FREE_SELECTIONS' * EXPORTING * it_tabname = lt_tabname * IMPORTING * ev_cancel = ev_cancel * et_whereclause = lt_where_clause. *** ....```

9.	```*** * STEP 9: Export selection criteria * *** CALL FUNCTION '/SCWM/SELOPT2RANGETAB' EXPORTING iv_repid = sy-repid it_mapping = lt_mapping IMPORTING et_tab_range = ct_tab_range. *** ....```
10.	```*** * STEP 10: Execute selection * *** TRY. lo_prd = /scwm/cl_dlv_management_prd=>get_instance( ). lo_prd->query(EXPORTING it_selection = lt_selection iv_doccat = /scdl/if_dl_doc_c=>sc_doccat_inb_prd is_read_options = ls_read_options is_include_data = ls_include_data IMPORTING et_headers = lt_head et_items = lt_item et_doctype_texts = lt_doctype_txt).```
Step	**Source Coding Snippets of Sample Function Module ZHTG_WHRHEAD_MON_XT**
	```CATCH /scdl/cx_delivery. "#EC NO_HANDLER ENDTRY. *** ....```
11.	```*** * STEP 11: Refine selection data * *** * Get status management instance ---------------------------------- --- IF go_stm IS INITIAL. TRY. go_stm = /scdl/cl_stm=>get_instance( ). CATCH /scdl/cx_stm. "#EC NO_HANDLER ENDTRY. ENDIF. * Get aggregation data --- --- REFRESH so_docid. REFRESH so_doci2. so_docid-sign = 'I'. so_docid- option = 'EQ'. so_doci2-sign = 'I'. so_doci2-option = 'EQ'. LOOP AT lt_head INTO ls_head. ....```

6.4 Form View for New Nodes (SAP S/4HANA 2022 or Later)

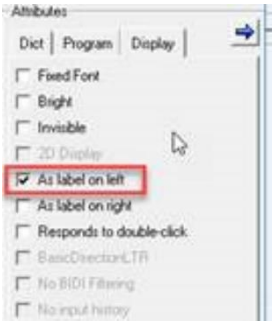
The warehouse management monitor offers form views for dedicated monitor nodes. In addition to the list view, a user can switch to a form view on parent level and child level. The basis for the form view is a standard dynpro. Perform the following steps to enable a form view for a monitor node:

1. Create a dynpro that displays a form view data in the function group of the node function module.
2. Add the fields of the node structure to the dynpro.

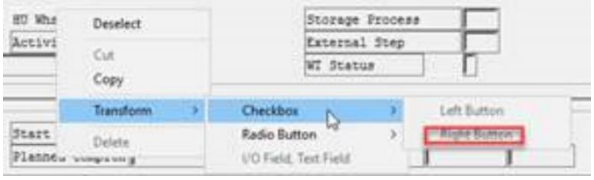


When adding the fields to the dynpro, we recommend the following settings:

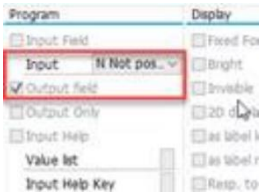
a. Add the field and the field label. Select the *As Label on Left* checkbox for the label screen element:



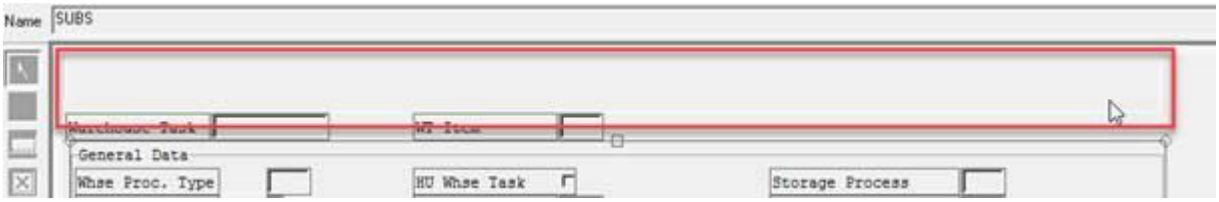
b. Ensure that the checkbox elements have a label on the left. Convert a checkbox if required:



c. Select the *Output Field* checkbox for the screen elements:



3. Add a subscreen area on the top of the form view dynpro as follows:
   a) Give the subscreen area the name *SUBS*:



b) Under *Resizing*, select the *Vertical* and *Horizontal* checkboxes and set the minimum rows to *2*. Specify the minimum column value depending on the width of the screen:

c) Adjust the master program of the function group where the form view dynpro is created. Also add include `/SCWM/RFORMVIEW_FORMVIEW`. This include contains central PBO/PAI module implementations.

d) Add the following implementation to the form view dynpro flow logic. This allows the use of central PBO/PAI modules for all form views:

Source Code of Form View Dynpro Flow Logic

```
PROCESS BEFORE OUTPUT.
  MODULE status_form_view.
  CALL SUBSCREEN subs INCLUDING '/SCWM/R_FORMVIEW' '1000'.

PROCESS AFTER INPUT.
  MODULE user_command_form_view.
  CALL SUBSCREEN subs.
```

e) Maintain the form view screen number and program in Customizing for *Extended Warehouse Management* under *Monitoring -> Warehouse Management Monitor -> Define Nodes*. Open the *Defined Nodes Profiles* folder and maintain *Dynpro Program* and *Dynpro No* for the node profile.

7. Maintain Hotspots for Navigation

As mentioned in chapter one, it is possible to specify hotspots. For example, SAP provides services for jumping directly to the warehouse order, warehouse task, and wave among others. Hotspot fields in the ALV output are underlined. The user can simply click on such a field to call the corresponding UI. Besides that, it is possible to mark multiple lines in the ALV and jump to these objects by using the jigsaw icon ( ) on the toolbar. Only Customizing entries have to be maintained to allow hotspots. The corresponding views can be found under the Customizing activity *Extended Warehouse Management → Monitoring → Warehouse Management Monitor → Define Navigation*.

Figure: Customizing Activity for Defining Hotspots

7.1 Example: Hotspot for Displaying Outbound Delivery Order Items in Enhanced Monitor Node

The SAP standard node for outbound delivery order items allows users to navigate to the corresponding outbound delivery by marking the item entry in the result list and double-clicking on the hotspot in the column *Document*:

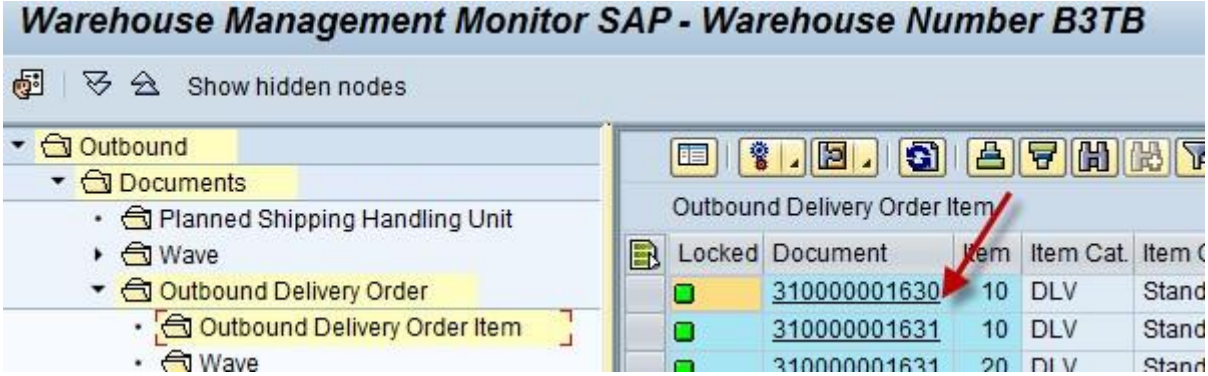


Figure: Hotspot Navigation in SAP Standard Monitor

The definition of a hotspot for a column of the result list is closely related to the corresponding list table type that belongs to the node profile of the monitor node. The technical name of hotspot is equal to the field name of the DDIC structure that is used in the list table type. In our example hotspot for navigation to outbound delivery, it is named *DOCNO_H* because the field *DOCNO_H* of the DDIC structure */SCWM/S_WIP_WHRITEM_OUT* defines the technical properties of the column *Document*.

Because the extended custom monitor uses a different list DDIC structure (structure *ZHTG_S_WIP_WHRITEM_OUT* instead of */SCWM/S_WIP_WHRITEM_OUT*), the hotspots were not copied from the template and have to be maintained in Customizing.

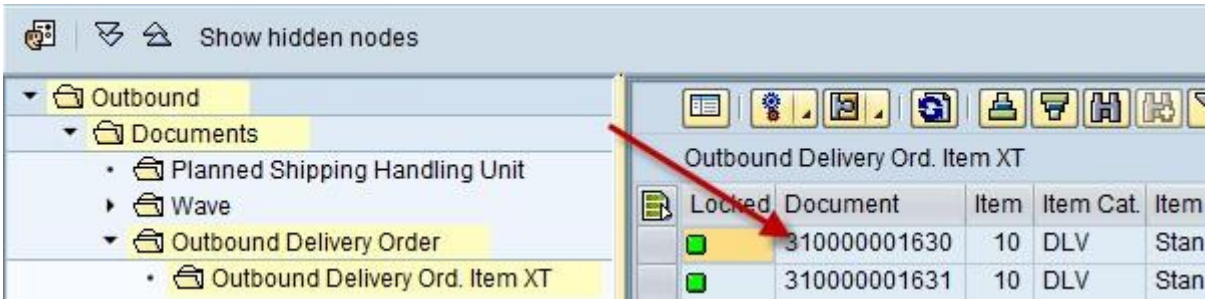


Figure: Missing Hotspot for Custom Node

Note

The hotspot *DOCNO_H* for the service DLV and list structure */SCWM/S_WIP_WHRITEM_OUT* is used as a template for creation of a similar hotspot for the custom node.

7.1.1 Define Hotspot

Execute the Customizing Activity *Define Navigation*. On the following screen double-click the folder *Define Hotspot*. Maintain the entries as described in the following table and save your changes:

List Structure	Hotspot	Service	Presentation Text
ZHTG_S_WIP_WHRITEM_OUT	DOCNO_H	DLV	

Important

The name of the hotspot must be the same as the name of the field in the DDIC structure that is used for creation of the result list.

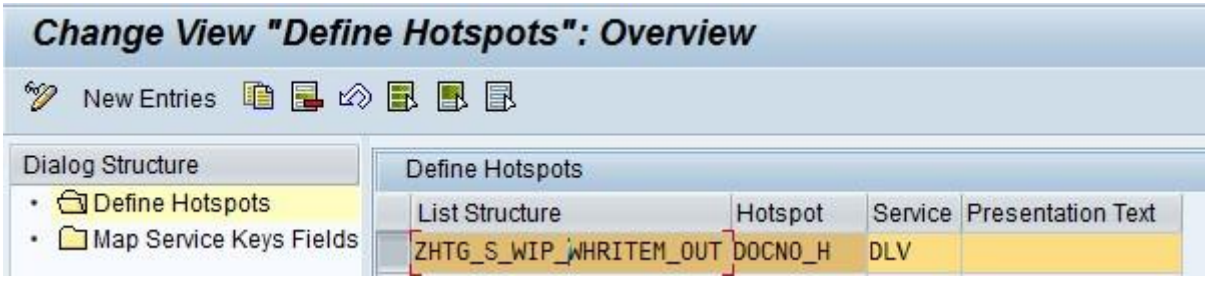


Figure: Creation of Hotspot for Custom Node

7.1.1.1 Involved Customizing Tables and Maintenance Views

Hotspot data is stored in the following tables:

- */SCWM/THOTSPOT* Hotspots
- */SCWM/THOTSPOTT* Hotspots Presentation Texts Corresponding maintenance view:
- */SCWM/V_HOTSPOT* Hotspots

Relevant field values of the Customizing table */SCWM/TOBCLS_MTH*:

Database Table	Field	Data Element	Description
/SCWM/THOTSPOT	LIST_ST	/SCWM/DE_LIST_ST	Object Structure for List View
/SCWM/THOTSPOT	HOTSPOT	/SCWM/DE_HOTSPOT	Hotspot Field
/SCWM/THOTSPOT	SERVICE	/SCWM/DE_SERVICE	Service
/SCWM/THOTSPOTT	PTEXT	/SCWM/DE_PTEXT	Presentation Text

7.1.2 Map Service Keys Fields

Using outbound deliveries as navigation targets requires that the outbound delivery number and document category (DOCCAT) are transferred. Therefore two entries have to be created in the mapping table for service keys fields. To do this, double-click on the folder *Map Service Keys Fields*. Maintain entries as described in the following table and save your changes:

List Structure	Hotspot	Service	Key Field	List View Field
ZHTG_S_WIP_WHRITEM_OUT	DOCNO_H	DLV	DOCCAT	DOCCAT
ZHTG_S_WIP_WHRITEM_OUT	DOCNO_H	DLV	DOCNO	DOCNO_H

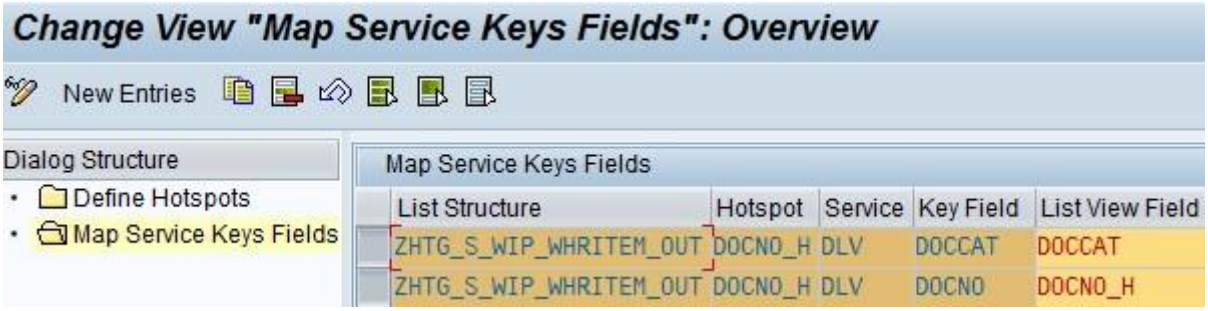


Figure: mapping Service Keys Fields for Hotspot ZHTG_DOCNO_H

Hotspot navigation is enabled after restarting the custom warehouse management monitor *ZHTG*:

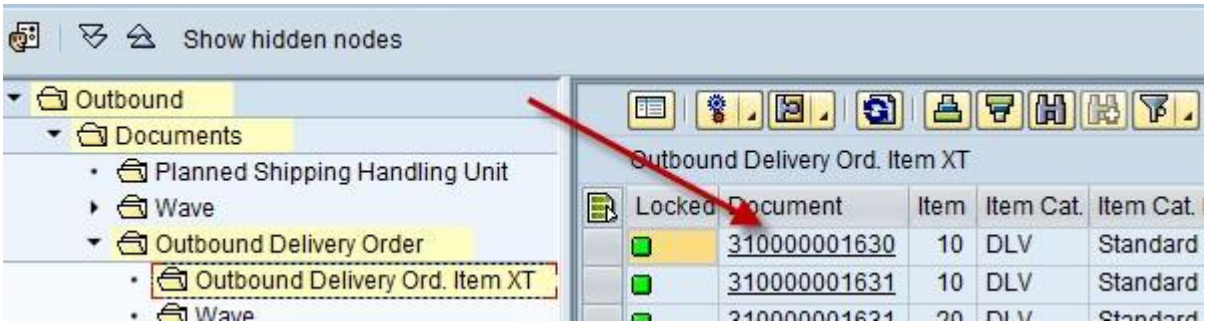


Figure: Custom Node with Active Hotspot for Column "Document"

7.1.2.1 Involved Customizing Tables and Maintenance Views

Tree hierarchy data is stored in the following table:

- */SCWM/TFLD_MAP* List view structure mapping Corresponding maintenance view:
- */SCWM/V_FLD_MAP* List view structure mapping

Relevant field values of the Customizing table */SCWM/TFLD_MAP*:

Field	Data Element	Description
LIST_ST	/SCWM/DE_LIST_ST	Object Structure for List View
HOTSPOT	/SCWM/DE_HOTSPOT	Hotspot Field Name
SERVICE	/SCWM/DE_SERVICE	Service to be used (for example, WO for warehouse orders)
KEYFIELD	/SCWM/DE_KEYFIELD	Key Field Name the service uses to identify the object
LIST_FIELD	/SCWM/DE_LIST_FIELD	Name of Field in Output List (List View Structure Field)
CONST_VAL	/SCWM/DE_MON_CONST_VAL	Fills KEYFIELD with a constant value (Constant Value for Monitor Navigation). Acts as alternative to field LIST_FIELD: If field LIST_FIELD is filled this field will be ignored

7.2 Example: Maintain Hotspot Navigation to Inbound Delivery

7.2.1 Define Hotspot

Execute the Customizing activity *Define Navigation*. On the following screen double-click the folder *Define Hotspots*. Maintain the entries as described in the following table and save your changes:

List Structure	Hotspot	Service	Presentation Text
ZHTG_S_WIP_WHRHEAD	DOCNO_H	DLV	

7.2.2 Map Service Keys Fields

Using inbound deliveries as navigation targets requires that the inbound delivery number and document category (DOCCAT) are transferred. As a consequence two entries have to be created in the mapping table for service keys fields. To do this, double-click on the folder *Map Service Keys Fields*. Maintain entries as described in the following table and save your changes:

List Structure	Hotspot	Service	Key Field	List View Field	Constant Value
ZHTG_ S_WIP_WHRHEAD	DOCNO_H	DLV	DOCCAT		PDI
ZHTG_ S_WIP_WHRHEAD	DOCNO_H	DLV	DOCNO	DOCNO_H	

8. Appendix

8.1 Sample Coding

8.1.1 Master Program SAPLZHTG (Function Pool)

Source Coding of Master Program SAPLZHTG

```
*********************************************************************
*     System-defined Include-files.                                 *
*********************************************************************
  INCLUDE lzhtgtop.                            " Global Data
  INCLUDE lzhtguxx.                            " Function Modules
*********************************************************************
*     User-defined Include-files (if necessary).                    *
*********************************************************************
*     INCLUDE LZHTGF...                            " Subroutines
*     INCLUDE LZHTGO...                            " PBO-Modules
*     INCLUDE LZHTGI...                            " PAI-Modules
*     INCLUDE LZHTGE...                            " Events
*     INCLUDE LZHTGP...                            " Local class definition.
*     INCLUDE LZHTGT99.                            " ABAP Unit tests
*     Local helper class (implementation)    INCLUDE lzhtgp02.
```

8.1.2 Include LZHTGTOP

Source Coding of Include LZHTGTOP

```
FUNCTION-POOL zhtg.                          "MESSAGE-ID ..

TYPES:
  BEGIN OF lty_docid_tu,
    docid       TYPE  /scdl/dl_docid,      tu_num_ext
TYPE  /scwm/de_tu_num_ext,   END OF lty_docid_tu,
lty_t_docid_tu TYPE STANDARD TABLE OF lty_docid_tu.

* Local helper class
definition INCLUDE lzhtgp01.
*********************************************************************
* Copied from /SCWM/LWIP_DELIVERYTOP
*********************************************************************
TABLES:
  /scwm/s_wip_q_whr_inbound,
  /scwm/s_sp_q_head,
sscrfields.
DATA:
  gv_dsrc   TYPE /scdl/dl_data_source VALUE 'DB'.            "#EC NEEDED
DATA:
  go_hlp_id TYPE REF TO lcl_hlp_id,                          "#EC NEEDED
```

Source Coding of Include LZHTGTOP

```
  go_stm    TYPE REF TO /scdl/cl_stm.                          "#EC NEEDED
CONSTANTS:
  gc_package_size  TYPE  i  VALUE 50.

* Dynpro 100 - Delivery header selection ------------------------------
SELECTION-SCREEN:
   BEGIN OF SCREEN 100 AS WINDOW,
   BEGIN OF BLOCK frm101 WITH FRAME TITLE text-001,
   BEGIN OF BLOCK bl110.
SELECTION-SCREEN BEGIN OF BLOCK frm000.
PARAMETERS: p_lgnum TYPE /scwm/s_wip_q_whr_inbound-lgnum NO-DISPLAY.
SELECTION-SCREEN END OF BLOCK frm000.
SELECT-OPTIONS:
  so_docno  FOR  /scwm/s_wip_q_whr_inbound-docno_h,
so_docty  FOR  /scwm/s_wip_q_whr_inbound-doctype_h,
***********************************************************************
* Added additional select-option for ERP documents
***********************************************************************
so_erpid  FOR  /scwm/s_sp_q_head-refdocno_erp_i,
***********************************************************************
so_asn    FOR  /scwm/s_wip_q_whr_inbound-refdocno_asn_h,   so_pro    FOR
/scwm/s_wip_q_whr_inbound-/scwm/refdocno_pro_h,       so_bol         FOR
/scwm/s_wip_q_whr_inbound-refdocno_bol_h,            so_tcdrf        FOR
/scwm/s_wip_q_whr_inbound-refdocno_tcd_h,           so_manu          FOR
/scwm/s_wip_q_whr_inbound-manual_h.
SELECTION-SCREEN:
  END OF BLOCK bl110.
SELECT-OPTIONS:
  so_huno   FOR  /scwm/s_wip_q_whr_inbound-huno.
SELECTION-SCREEN:
   BEGIN OF BLOCK bl111.
SELECT-OPTIONS:   so_tu     FOR  /scwm/s_wip_q_whr_inbound-
transmeans_id_h,   so_tu_ex  FOR
/scwm/s_wip_q_whr_inbound-tu. SELECTION-SCREEN:
  BEGIN OF LINE,
  COMMENT 1(31) text-007 FOR FIELD p_tudfr.
PARAMETERS:
  p_tudfr   TYPE   /scwm/s_wip_q_whr_inbound-tu_date_from,
p_tutfr    TYPE    /scwm/s_wip_q_whr_inbound-tu_time_from.
SELECTION-SCREEN COMMENT 55(20) text-003 FOR FIELD p_tudto.
PARAMETERS:
  p_tudto  TYPE  /scwm/s_wip_q_whr_inbound-tu_date_to,
p_tutto  TYPE  /scwm/s_wip_q_whr_inbound-tu_time_to.
SELECTION-SCREEN END OF LINE.
SELECT-OPTIONS:
  so_door      FOR    /scwm/s_wip_q_whr_inbound-/scwm/door_i,
so_carr      FOR    /scwm/s_wip_q_whr_inbound-partyno_carr_h,
so_sfprt    FOR    /scwm/s_wip_q_whr_inbound-partyno_sfprt_h,
so_stprf    FOR    /scwm/s_wip_q_whr_inbound-partyno_stprtf_h,
so_sflo     FOR   /scwm/s_wip_q_whr_inbound-locationno_sflo_h,
so_dwai     FOR   /scwm/s_wip_q_whr_inbound-status_value_dwa_i,
so_dtrh     FOR   /scwm/s_wip_q_whr_inbound-status_value_dtr_h,
so_duni     FOR   /scwm/s_wip_q_whr_inbound-status_value_dun_i,
so_dgri     FOR   /scwm/s_wip_q_whr_inbound-status_value_dgr_i,
so_deui     FOR   /scwm/s_wip_q_whr_inbound-status_value_deu_i,
so_dpti     FOR   /scwm/s_wip_q_whr_inbound-status_value_dpt_i,
so_dwnh   FOR  /scwm/s_wip_q_whr_inbound-status_value_dwn_h.
* dynamic status (item)
```

Source Coding of Include LZHTGTOP

```
SELECTION-SCREEN:
  BEGIN OF LINE,
  COMMENT 1(31) text-020 FOR FIELD po_dsity.
PARAMETERS     po_dsity TYPE /scwm/s_wip_q_whr_inbound-status_type_item.
SELECT-OPTIONS so_dsiva FOR /scwm/s_wip_q_whr_inbound-status_value_item.
SELECTION-SCREEN END OF LINE. * locked item
PARAMETERS:
  p_dboi   TYPE  /scwm/s_wip_q_whr_inbound-status_value_dbo_i.
* Delivery date SELECTION-SCREEN:
  BEGIN OF BLOCK bl102,
  BEGIN OF LINE,
  COMMENT 1(31) gv_txtdt FOR FIELD p_dlvdfr.
PARAMETERS:
  p_dlvdfr  TYPE  /scwm/s_wip_whrhead-dlv_date,
p_dlvtfr  TYPE  /scwm/s_wip_whrhead-dlv_time.
SELECTION-SCREEN COMMENT 55(20) text-003 FOR FIELD p_dlvdto.
PARAMETERS:
  p_dlvdto  TYPE  /scwm/s_wip_whrhead-dlv_date,
p_dlvtto  TYPE  /scwm/s_wip_whrhead-dlv_time.
SELECTION-SCREEN:
  END OF LINE,
  END OF BLOCK bl102.
SELECT-OPTIONS:   so_dlvdt FOR /scwm/s_wip_q_whr_inbound-
tstfr_tdelivery_plan_h
                                                            NO-DISPLAY,
so_tudt    FOR /scwm/s_wip_q_whr_inbound-tstfr_tu           NO-DISPLAY,
so_docid   FOR /scwm/s_wip_q_whr_inbound-docid_h            NO-DISPLAY,
so_doci2   FOR /scwm/s_wip_q_whr_inbound-docid_h          NO-DISPLAY.
SELECTION-SCREEN:
  END OF BLOCK bl111,
END OF BLOCK frm101.

*Datasource block SELECTION-SCREEN:
  BEGIN OF BLOCK frm102.

SELECTION-SCREEN BEGIN OF BLOCK arch_o WITH FRAME TITLE text-030.
PARAMETERS:
  p_db   TYPE /scwm/s_sp_q_head-data_source_db RADIOBUTTON GROUP dsrc,
p_arch TYPE /scwm/s_sp_q_head-data_source_arch RADIOBUTTON GROUP dsrc,
p_both TYPE /scwm/s_sp_q_head-data_source_both RADIOBUTTON GROUP dsrc.
SELECTION-SCREEN END OF BLOCK arch_o.

SELECTION-SCREEN:
  END OF BLOCK frm102,
FUNCTION KEY 1,
  END OF SCREEN 100.
```

8.1.3 Include LZHTGP01

Source Coding of Sample Function Module LZHTGP01
`*&---------------------------------------------------------------------*` `*&  Include           LZHTGP01`

Source Coding of Sample Function Module LZHTGP01

```
*&---------------------------------------------------------------------*
CLASS lcl_hlp_id DEFINITION CREATE PUBLIC.

  PUBLIC SECTION.
    METHODS:
convert_date_time_to_ts
IMPORTING
          iv_date_from  TYPE datum
iv_date_to    TYPE datum
iv_lgnum      TYPE /scwm/lgnum
iv_time_from  TYPE uzeit
iv_time_to    TYPE uzeit         EXPORTING
          et_timestamp  TYPE /scwm/tt_timestamp_r,
convert_timestamp         IMPORTING
          iv_tstfr     TYPE  tzntstmps
iv_lgnum     TYPE  /scwm/lgnum
EXPORTING
          es_time_date TYPE
/scwm/s_tstmp_date_time,       fill_include_data
CHANGING
          cs_incl_data TYPE
/scwm/dlv_query_incl_str_prd,       fill_selection_table
IMPORTING
          iv_lgnum      TYPE  /scwm/lgnum
CHANGING
          ct_selection  TYPE
/scwm/dlv_selection_tab,       map_whrhead_to_sel_opt
CHANGING
          ct_mapping TYPE /scwm/tt_map_selopt2field,
modify_field_cat
        CHANGING ct_fieldcat TYPE lvc_t_fcat,
select_huref         CHANGING
          ct_aggr TYPE /scwm/tt_wip_whr_aggr,
select_tu         CHANGING
          ct_docid_tunum TYPE lty_t_docid_tu,
set_status_data         IMPORTING
          it_status TYPE /scdl/dl_status_tab
CHANGING           cs_whrhead TYPE
zhtg_s_wip_whrhead.

  PROTECTED SECTION.
    METHODS:
      add_sel_crit_common
IMPORTING
          iv_lgnum      TYPE  /scwm/lgnum
CHANGING
          ct_huident_r  TYPE  rseloption
          ct_selection  TYPE
/scwm/dlv_selection_tab,       add_sel_crit_status
IMPORTING
          iv_lgnum      TYPE  /scwm/lgnum
CHANGING
          ct_selection  TYPE  /scwm/dlv_selection_tab,
get_status_text
         IMPORTING
```

Source Coding of Sample Function Module LZHTGP01

```
            is_status_stm     TYPE   /scdl/stm_status_str              EXPORTING
es_status_text  TYPE  /scdl/stm_status_text_str.

  PRIVATE SECTION.

ENDCLASS.                    "lcl_hlp_id DEFINITION
```

8.1.4 Include LZHTGP02

Source Coding of Sample Function Module LZHTGP02

```
*&---------------------------------------------------------------------*
*&  Include           LZHTGP02
*&---------------------------------------------------------------------*
CLASS lcl_hlp_id IMPLEMENTATION.

*---------------------------------------------------------------------*
* Public methods
*---------------------------------------------------------------------*
  METHOD convert_date_time_to_ts.
*     IMPORTING
*     iv_date_from  TYPE datum
*     iv_date_to    TYPE datum
*     iv_lgnum      TYPE /scwm/lgnum
*     iv_time_from  TYPE uzeit
*     iv_time_to    TYPE uzeit
*     EXPORTING
*     et_timestamp  TYPE /scwm/tt_timestamp_r

    DATA:
      ls_date_time_from TYPE /scwm/s_date_time,
ls_date_time_to   TYPE /scwm/s_date_time,
ls_timestamp_r    TYPE /scwm/s_timestamp_r.

    CLEAR: et_timestamp.

    MOVE: iv_date_from TO ls_date_time_from-date,
iv_time_from TO ls_date_time_from-time,
iv_date_to   TO ls_date_time_to-date,
iv_time_to   TO ls_date_time_to-time.
    CALL FUNCTION '/SCWM/CONVERT_DATE_TIME'
EXPORTING
        iv_lgnum           = iv_lgnum
is_dattim_from     = ls_date_time_from
is_dattim_to       = ls_date_time_to
IMPORTING
        es_timestamp_range = ls_timestamp_r
EXCEPTIONS
        input_error        = 1
data_not_found     = 2
OTHERS             = 3.
    CASE sy-subrc.
      WHEN 0.
        IF ls_timestamp_r IS NOT INITIAL.
```

```
            APPEND ls_timestamp_r TO et_timestamp.
```

Source Coding of Sample Function Module LZHTGP02

```
        ENDIF.
      WHEN 1.
      WHEN OTHERS.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDCASE.

  ENDMETHOD.                    "convert_date_time_to_ts
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD convert_timestamp.
*     IMPORTING
*     iv_tstfr     TYPE  tzntstmps
*     iv_lgnum     TYPE  /scwm/lgnum
*     EXPORTING
*     es_time_date TYPE  /scwm/s_tstmp_date_time

    DATA:                     lt_timestamp      TYPE
/scwm/tt_timestamp,             lt_timedate     TYPE
/scwm/tt_tstmp_date_time.

    CLEAR:
      es_time_date.

    APPEND iv_tstfr TO lt_timestamp.

    CALL FUNCTION '/SCWM/CONVERT_TIMESTAMP'
EXPORTING
        iv_lgnum       = iv_lgnum
it_timestamp    = lt_timestamp       IMPORTING
        et_date_time    =
lt_timedate       EXCEPTIONS
input_error     = 1
data_not_found = 2         OTHERS
= 3.      IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

    READ TABLE lt_timedate INTO es_time_date INDEX 1.

  ENDMETHOD.                    "convert_timestamp
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD fill_include_data.
*     CHANGING
*     cs_incl_data TYPE  /scwm/dlv_query_incl_str_prd,
*     Fill exclude structure ------------------------------------------
----
    CLEAR cs_incl_data.
    cs_incl_data-head_partyloc    = abap_true.     cs_incl_data-
head_date        = abap_true.     cs_incl_data-head_status     =
abap_true.
    cs_incl_data-head_status_dyn
        = /scwm/if_dl_c=>sc_h_stat_dyn_calc_all_items.
```

Source Coding of Sample Function Module LZHTGP02

```
* For correct status calculation of DWA we need all related status added
APPEND /scdl/if_dl_status_c=>sc_t_warehouse_activity
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_planning_unloading
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_planning_putaway
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_goods_receipt
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_putaway
TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_split
TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_unloading
TO cs_incl_data-head_status_dyn_detail.
    APPEND /scwm/if_dl_qm_status_c=>sc_t_qi_execution
        TO cs_incl_data-head_status_dyn_detail.
APPEND /scwm/if_dl_status_c=>sc_t_vas
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_goods_receipt_dist
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_unloading_dist
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_plan_unload_dist
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_plan_putaway_dist
        TO cs_incl_data-head_status_dyn_detail.
    APPEND /scdl/if_dl_status_c=>sc_t_putaway_dist
TO cs_incl_data-head_status_dyn_detail.
     cs_incl_data-head_refdoc     =
abap_true.     cs_incl_data-head_transport
= abap_true.     cs_incl_data-item_partyloc
= abap_true.     cs_incl_data-item_date
= abap_true.

  ENDMETHOD.                    "fill_include_data
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD fill_selection_table.
*     IMPORTING
*     iv_lgnum      TYPE  /scwm/lgnum
*     CHANGING
*     ct_selection  TYPE  /scwm/dlv_selection_tab

    DATA:       lt_docid_tu      TYPE
/scwm/dlv_docid_itemid_tab,       lt_huident_r     TYPE
rseloption,       lt_huref         TYPE
/scwm/tt_huref_int,       lt_timestamp_r   TYPE
/scwm/tt_timestamp_r.
    DATA:                     ls_doccat_r                  TYPE
/scwm/s_sel_dlv_doccat,              ls_docid_tu            TYPE
/scwm/dlv_docid_itemid_str,           ls_huref              TYPE
/scwm/s_huref_int,                ls_select                 TYPE
/scwm/dlv_selection_str,             ls_t300_md             TYPE
/scwm/s_t300_md,                     ls_timestamp_r         TYPE
```

```
/scwm/s_timestamp_r,      ls_tu_ext       LIKE LINE OF  so_tu_ex,
ls_tu_r          TYPE          /scwm/s_sel_tu_num_ext.     DATA:
```

Source Coding of Sample Function Module LZHTGP02

```
      lv_timestamp_h   TYPE          timestamp,
lv_timestamp_l   TYPE          timestamp.
    DATA:       lo_log_wm        TYPE REF TO
/scwm/cl_log,       lo_query_tu      TYPE REF TO
/scwm/cl_sr_tu_query.

*   Warehouse: Get SC Unit location number
    CALL FUNCTION '/SCWM/T300_MD_READ_SINGLE'
EXPORTING
        iv_lgnum   = iv_lgnum
IMPORTING
        es_t300_md =
ls_t300_md       EXCEPTIONS
not_found  = 1         OTHERS
= 2.     IF sy-subrc EQ 0.
      CLEAR ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_locationid_wh_h.
ls_select-option    = 'EQ'.       ls_select-sign
= 'I'.       ls_select-low       = ls_t300_md-
scuguid.
      APPEND ls_select TO ct_selection.
    ELSE.
      MESSAGE text-006 TYPE 'E'.
*     MESSAGE text-006 TYPE 'E' RAISING no_sc_unit.     ENDIF.

    IF go_hlp_id IS INITIAL.
CREATE OBJECT go_hlp_id.
    ENDIF.

*     Planned delivery date
    go_hlp_id->convert_date_time_to_ts(
EXPORTING
        iv_date_from = p_dlvdfr
iv_date_to   = p_dlvdto
iv_lgnum     = iv_lgnum
iv_time_from = p_dlvtfr
iv_time_to   = p_dlvtto       IMPORTING
et_timestamp = lt_timestamp_r ).

    LOOP AT lt_timestamp_r INTO ls_timestamp_r.
      CLEAR ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_tstfr_tdelivery_plan_h.
      ls_select-option    = ls_timestamp_r-option.
ls_select-low       = ls_timestamp_r-low.       ls_select-
high      = ls_timestamp_r-high.
      APPEND ls_select TO ct_selection.
    ENDLOOP.

*   Add common selection criteria
go_hlp_id->add_sel_crit_common(
EXPORTING
        iv_lgnum = iv_lgnum
CHANGING
        ct_huident_r = lt_huident_r
ct_selection = ct_selection ).
```

Source Coding of Sample Function Module LZHTGP02

```
*     Add selection criteria for status     go_hlp_id-
>add_sel_crit_status(       EXPORTING
        iv_lgnum = iv_lgnum
CHANGING         ct_selection =
ct_selection ).

*     Preselection if HU number was a search criterion ----
-----------------     IF NOT lt_huident_r IS INITIAL.

      CALL FUNCTION '/SCWM/HU_SELECT_GEN'
EXPORTING
          iv_lgnum     = iv_lgnum
ir_huident   = lt_huident_r
IMPORTING
          et_huref     =
lt_huref         EXCEPTIONS
wrong_input  = 1
not_possible = 2
OTHERS       = 3.       IF sy-
subrc <> 0.
        MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
          WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-
msgv4.       ENDIF.
      DELETE lt_huref WHERE doccat NE /scdl/if_dl_c=>sc_doccat_inb_prd.
      IF lt_huref IS INITIAL.
*       No data for specified HU => Clear sel. table and add dummy DOCID
        CLEAR:
          ct_selection,           ls_select.         ls_select-
fieldname = /scdl/if_dl_logfname_c=>sc_docid_h.
ls_select-option    = /scmb/cl_search=>sc_eq.         ls_select-
low       = 'NOT_EXISTENT'.
        APPEND ls_select TO ct_selection.
        RETURN.
      ELSE.
        LOOP AT lt_huref INTO ls_huref.
          CLEAR ls_select.
          ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_docid_h.
          ls_select-option    = /scmb/cl_search=>sc_eq.
ls_select-low       = ls_huref-docid.
          APPEND ls_select TO ct_selection.
        ENDLOOP.
      ENDIF.
    ENDIF.
*     Preselection if TU number was a search criterion -------------------
--
    IF NOT so_tu_ex[] IS INITIAL.
*     Create required instances for TU query
      CREATE OBJECT lo_log_wm.
      CREATE OBJECT
lo_query_tu         EXPORTING
io_log = lo_log_wm.

*     Set DOCCAT for query       ls_doccat_r-sign    =
/scmb/cl_search=>sc_sign_i.       ls_doccat_r-option =
/scmb/cl_search=>sc_eq.       ls_doccat_r-low     =
/scdl/if_dl_doc_c=>sc_doccat_inb_prd.
```

Source Coding of Sample Function Module LZHTGP02

```
      lo_query_tu->add_dlv_doccat(
        EXPORTING is_sel_dlv_doccat = ls_doccat_r ).

*     Fill TU_NUM_EXT to query       LOOP AT so_tu_ex INTO
ls_tu_ext.
        MOVE-CORRESPONDING ls_tu_ext TO ls_tu_r.
        lo_query_tu->add_tu_num_ext(
          EXPORTING is_tu_num_ext = ls_tu_r ).
ENDLOOP.
*
*     Start query with time selection range
      CLEAR: lt_timestamp_r[].       go_hlp_id-
>convert_date_time_to_ts(         EXPORTING
          iv_date_from = p_tudfr
iv_date_to    = p_tudto           iv_lgnum
= iv_lgnum           iv_time_from =
p_tutfr           iv_time_to    = p_tutto
IMPORTING           et_timestamp =
lt_timestamp_r ).

      CLEAR:
ls_timestamp_r,
lt_docid_tu.
      READ TABLE lt_timestamp_r INTO ls_timestamp_r INDEX
1.       lv_timestamp_l = ls_timestamp_r-low.
lv_timestamp_h = ls_timestamp_r-high.

      CALL FUNCTION '/SCWM/GET_DLV_BY_TU_QUERY'
EXPORTING
          io_tu_query    = lo_query_tu
iv_sel_start   = lv_timestamp_l
iv_sel_end     = lv_timestamp_h
iv_do_refresh = abap_true
          iv_call_mode   =
wmesr_call_mode_read_only         IMPORTING
et_docid       = lt_docid_tu.

*     Adjust selection table       IF lt_docid_tu IS
INITIAL.
*     No data for specified TU => Clear sel. table and
add dummy DOCID
        CLEAR:
          ct_selection,           ls_select.         ls_select-
fieldname = /scdl/if_dl_logfname_c=>sc_docid_h.
ls_select-option     = /scmb/cl_search=>sc_eq.
        ls_select-low        = 'NOT_EXISTENT'.
        APPEND ls_select TO ct_selection.
        RETURN.
      ELSE.
        SORT lt_docid_tu BY docid.
        DELETE ADJACENT DUPLICATES FROM lt_docid_tu COMPARING docid.
        LOOP AT lt_docid_tu INTO ls_docid_tu.
          CLEAR ls_select.
          ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_docid_h.
ls_select-option     = /scmb/cl_search=>sc_eq.           ls_select-
low        = ls_docid_tu-docid.
          APPEND ls_select TO ct_selection.
ENDLOOP.
```

Source Coding of Sample Function Module LZHTGP02

```
      ENDIF.
    ENDIF.

*     Finally specify the datasource (database, archive)
    IF p_db IS INITIAL.
CLEAR ls_select.
*     If datasource is DB then we don't do anything -> default behaviour
      IF p_arch IS NOT INITIAL.         ls_select-
fieldname =
          /scwm/if_ui_c=>sc_field_data_source_arch.
ls_select-sign  = /scmb/cl_search=>sc_sign_i.
ls_select-option    = /scmb/cl_search=>sc_eq.
ls_select-low       = abap_true.         gv_dsrc =
/scdl/if_dl_query_c=>sc_source_archive.
      ELSEIF p_both IS NOT INITIAL.
        ls_select-fieldname =
          /scwm/if_ui_c=>sc_field_data_source_both.
ls_select-sign  = /scmb/cl_search=>sc_sign_i.         ls_select-
option    = /scmb/cl_search=>sc_eq.
        ls_select-low       = abap_true.
        gv_dsrc = /scdl/if_dl_query_c=>sc_source_both.
      ELSE.
*       Both parameters are initial, use the global variable
*       it happens during navigation (header -> item)
        CASE gv_dsrc.
          WHEN /scdl/if_dl_query_c=>sc_source_both.
ls_select-fieldname =
              /scwm/if_ui_c=>sc_field_data_source_both.
ls_select-sign  = /scmb/cl_search=>sc_sign_i.             ls_select-
option    = /scmb/cl_search=>sc_eq.             ls_select-low       =
abap_true.
          WHEN /scdl/if_dl_query_c=>sc_source_archive.
            ls_select-fieldname =
              /scwm/if_ui_c=>sc_field_data_source_arch.
ls_select-sign  = /scmb/cl_search=>sc_sign_i.             ls_select-
option    = /scmb/cl_search=>sc_eq.
            ls_select-low       = abap_true.
        ENDCASE.
      ENDIF.
      IF ls_select IS NOT INITIAL.
APPEND ls_select TO ct_selection.
      ENDIF.
    ENDIF.

  ENDMETHOD.                    "fill_selection_table
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD map_whrhead_to_sel_opt.
*   CHANGING ct_mapping TYPE /scwm/tt_map_selopt2field
    DATA:
      ls_mapping  TYPE  /scwm/s_map_selopt2field.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DOCNO'                    TO ls_mapping-selname,
          'DOCNO_H'                    TO ls_mapping-fieldname,
abap_true                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
CLEAR ls_mapping.
```

Source Coding of Sample Function Module LZHTGP02

```
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DOCTY'                    TO ls_mapping-selname,
          'DOCTYPE_H'                   TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_TCDREF'                   TO ls_mapping-selname,
          'REFDOCNO_TCD_H'             TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_BOL'                      TO ls_mapping-selname,
          'REFDOCNO_BOL_H'             TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_PPO'                      TO ls_mapping-selname,
          'REFDOCNO_PPO_I'             TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_MANUAL'                   TO ls_mapping-selname,
          'MANUAL_H'                   TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_TU'                       TO ls_mapping-selname,
          'TRANSMEANS_ID_H'            TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_HUNO'                     TO ls_mapping-selname,
          'HUNO'                        TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DOOR'                     TO ls_mapping-selname,
          '/SCWM/DOOR_I'               TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_ASN'                      TO ls_mapping-selname,
          'REFDOCNO_ASN_H'             TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_PRO'                      TO ls_mapping-selname,
```

```
          '/SCWM/REFDOCNO_PRO_H'        TO ls_mapping-fieldname,
abap_false                     TO ls_mapping-is_key.
    APPEND ls_mapping                   TO ct_mapping.
CLEAR ls_mapping.
```

Source Coding of Sample Function Module LZHTGP02

```
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_SFLO'                     TO ls_mapping-selname,
          'LOCATIONNO_SFLO_H'          TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_STPRTF'                   TO ls_mapping-selname,
          'PARTYNO_STPRTF_H'           TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_SFPRT'                    TO ls_mapping-selname,
          'PARTYNO_SFPRT_H'            TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_CARR'                     TO ls_mapping-selname,
          'PARTYNO_CARR_H'             TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DTRH'                     TO ls_mapping-selname,
          'STATUS_VALUE_DTR_H'         TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DGRI'                     TO ls_mapping-selname,
          'STATUS_VALUE_DGR_I'         TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DPTI'                     TO ls_mapping-selname,
          'STATUS_VALUE_DPT_I'         TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'P_DBOI'                     TO ls_mapping-selname,
          'STATUS_VALUE_DBO_I'         TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_TU_EX'                    TO ls_mapping-selname,
          'TU'                         TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DEUI'                     TO ls_mapping-selname,
```

```
          'STATUS_VALUE_DEU_I'          TO ls_mapping-fieldname,
abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                   TO ct_mapping.
CLEAR ls_mapping.
```

Source Coding of Sample Function Module LZHTGP02

```
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DUNI'                     TO ls_mapping-selname,
          'STATUS_VALUE_DUN_I'         TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DWAI'                     TO ls_mapping-selname,
          'STATUS_VALUE_DWA_I'         TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DWNH'                     TO ls_mapping-selname,
          'STATUS_VALUE_DWN_H'         TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_DLVDT'                    TO ls_mapping-selname,
          'TSTFR_TDELIVERY_PLAN_H'        TO  ls_mapping-fieldname,
abap_true                             TO  ls_mapping-is_timestamp,
'P_DLVDFR'                   TO ls_mapping-p_date_from,
          'P_DLVTFR'                   TO ls_mapping-p_time_from,
          'P_DLVDTO'                   TO ls_mapping-p_date_to,
          'P_DLVTTO'                   TO ls_mapping-p_time_to,
          'DLV_DATE'                   TO ls_mapping-date_field,
'DLV_TIME'                   TO ls_mapping-time_field.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'S_TUDT'                     TO ls_mapping-selname,
          'TSTFR_TU'                      TO ls_mapping-fieldname,
abap_true                             TO  ls_mapping-is_timestamp,
'P_TUDFR'                    TO ls_mapping-p_date_from,
          'P_TUTFR'                    TO ls_mapping-p_time_from,
          'P_TUDTO'                     TO ls_mapping-p_date_to,
'P_TUTTO'                 TO ls_mapping-p_time_to,          space
TO ls_mapping-date_field,          space                       TO
ls_mapping-time_field.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'PO_DSITY'                   TO ls_mapping-selname,
          'STATUS_TYPE_ITEM'           TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_WIP_Q_WHR_INBOUND'  TO ls_mapping-tablename,
          'SO_DSIVA'                   TO ls_mapping-selname,
          'STATUS_VALUE_ITEM'          TO ls_mapping-fieldname,
abap_false                   TO ls_mapping-is_key.
    APPEND ls_mapping                  TO ct_mapping.
*********************************************************************
* Add additional selection criteria for ERP documents               *
*********************************************************************
    CLEAR ls_mapping.
    MOVE: '/SCWM/S_SP_Q_HEAD'          TO ls_mapping-tablename,
```

```
        'SO_ERPID'                    TO ls_mapping-selname,
        'REFDOCNO_ERP_I'              TO ls_mapping-fieldname,
```

Source Coding of Sample Function Module LZHTGP02

```
          abap_false                    TO ls_mapping-is_key.
    APPEND ls_mapping                   TO ct_mapping.
*********************************************************************
ENDMETHOD.                        "map_whrhead_to_sel_opt
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD modify_field_cat.
*     CHANGING
*     ct_fieldcat TYPE lvc_t_fcat

    DATA:
      lt_fcat          TYPE lvc_t_fcat.
DATA:
      ls_fcat          TYPE lvc_s_fcat.
    DATA:
      lv_idx_docid     TYPE sy-tabix.

    CONSTANTS:       lc_fld_docid     TYPE lvc_fname
VALUE 'DOCID',       lc_fld_erp_docno TYPE lvc_fname
VALUE 'ERP_DOCNO'.

*     Changing field catalogue CT_FIELDCAT: Change positions of ERP
*     document number. This column shall appear behind column containing
*     document ID (DOCID).
    LOOP AT ct_fieldcat INTO
ls_fcat.       CASE ls_fcat-
fieldname.         WHEN
lc_fld_docid.
*     Store index of field cat entry for column 'DOCID'           lv_idx_docid
= sy-tabix.
        WHEN lc_fld_erp_docno.
*     Display ERP document number after column containing document
*     ID           ls_fcat-col_pos = lv_idx_docid + 1.
        WHEN OTHERS.
          IF NOT lv_idx_docid IS INITIAL.
*     Columns positioned after column DOCID: Change position
ls_fcat-col_pos = ls_fcat-col_pos + 2.
          ENDIF.
      ENDCASE.
      INSERT ls_fcat INTO TABLE lt_fcat.
    ENDLOOP.

    IF NOT lt_fcat[] IS INITIAL.
*     Transfer back interim results
      CLEAR ct_fieldcat[].
      ct_fieldcat[] = lt_fcat[].
    ENDIF.

  ENDMETHOD.                      "modify_field_cat
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD select_huref.
*     CHANGING ct_aggr TYPE /scwm/tt_wip_whr_aggr

    DATA:
```

```
      lt_aggr  TYPE  /scwm/tt_wip_whr_aggr.
*     Select from DB
```

Source Coding of Sample Function Module LZHTGP02

```
    SELECT docid COUNT( DISTINCT guid_hu ) AS no_hu
      FROM /scwm/huref
      INTO CORRESPONDING FIELDS OF TABLE lt_aggr
      WHERE docid IN so_doci2
        AND doccat = /scdl/if_dl_doc_c=>sc_doccat_inb_prd
GROUP BY docid.

*     Store result
    APPEND LINES OF lt_aggr TO ct_aggr.

*     Sort data, delete duplicates     SORT ct_aggr BY docid.
    DELETE ADJACENT DUPLICATES FROM ct_aggr COMPARING docid.

  ENDMETHOD.                    "select_huref
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD select_tu.
*     CHANGING ct_docid_tunum TYPE lty_t_docid_tu.

    DATA:
      lt_docid_tunum  TYPE  lty_t_docid_tu.

    SELECT d~docid t~tu_num_ext
      INTO CORRESPONDING FIELDS OF TABLE lt_docid_tunum
      FROM       /scwm/tu_dlv AS d
      INNER JOIN /scwm/tunit  AS t
        ON t~tu_num = d~tu_num
WHERE d~docid IN so_doci2.

*     Store result
    APPEND LINES OF lt_docid_tunum TO ct_docid_tunum.

*     Sort data, delete duplicates
    SORT ct_docid_tunum BY docid tu_num_ext.
    DELETE ADJACENT DUPLICATES FROM ct_docid_tunum
COMPARING docid tu_num_ext.

  ENDMETHOD.                    "select_tu
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD set_status_data.
*     IMPORTING
*     it_status  TYPE /scdl/dl_status_tab
*     CHANGING
*     cs_whrhead TYPE zhtg_s_wip_whrhead.

    DATA:
      ls_status        TYPE         /scdl/dl_status_str,
ls_status_stm    TYPE         /scdl/stm_status_str,
ls_status_text   TYPE         /scdl/stm_status_text_str.

    LOOP AT it_status INTO ls_status.
CASE ls_status-status_type.
        WHEN /scdl/if_dl_c=>sc_t_in_yard.
```

```
          ls_status_stm-status_type      =   /scdl/if_dl_c=>sc_t_in_yard.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
```

Source Coding of Sample Function Module LZHTGP02

```
            EXPORTING
              is_status_stm = ls_status_stm             IMPORTING
es_status_text =  ls_status_text ).           cs_whrhead-
status_inyard = ls_status_text-status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_unloading.
          ls_status_stm-status_type     =  /scdl/if_dl_c=>sc_t_unloading.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_unloading =             ls_status_text-
status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_warehouse_activity.
          ls_status_stm-status_type  =
            /scdl/if_dl_c=>sc_t_warehouse_activity.
          ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_wm_activity =             ls_status_text-
status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_goods_receipt.
          ls_status_stm-status_type = /scdl/if_dl_c=>sc_t_goods_receipt.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_gr = ls_status_text-status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_putaway.
          ls_status_stm-status_type = /scdl/if_dl_c=>sc_t_putaway.
          ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_putaway = ls_status_text-status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_planning_putaway.
          ls_status_stm-status_type =
            /scdl/if_dl_c=>sc_t_planning_putaway.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
```

```
          cs_whrhead-status_putaway_plan =             ls_status_text-
status_value_text.
```

Source Coding of Sample Function Module LZHTGP02

```
        WHEN /scwm/if_dl_c=>sc_t_ncts.
ls_status_stm-status_type =
/scwm/if_dl_c=>sc_t_ncts.           ls_status_stm-status_value
= ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm             IMPORTING
es_status_text =  ls_status_text ).           cs_whrhead-
status_ncts = ls_status_text-status_value_text.

        WHEN /scdl/if_dl_c=>sc_t_blocked_overall.
          CALL FUNCTION '/SCWM/WIP_GET_DLV_STATUS_ICON'
EXPORTING
              iv_status_type  = /scdl/if_dl_c=>sc_t_blocked_overall
              iv_status_value = ls_status-status_value
IMPORTING               ev_icon         = cs_whrhead-
status_blocked_overall.

        WHEN /scdl/if_dl_status_c=>sc_t_goods_receipt_dist.
ls_status_stm-status_type = ls_status-status_type.           ls_status_stm-
status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).           cs_whrhead-
status_goods_receipt_dist             =
ls_status_text-status_value_text.

        WHEN /scdl/if_dl_status_c=>sc_t_unloading_dist.
          ls_status_stm-status_type = ls_status-status_type.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_unloading_dist
= ls_status_text-status_value_text.

        WHEN /scdl/if_dl_status_c=>sc_t_plan_unload_dist.
ls_status_stm-status_type = ls_status-status_type.           ls_status_stm-
status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING               es_status_text =
ls_status_text ).
          cs_whrhead-status_plan_unload_dist
            = ls_status_text-status_value_text.

        WHEN /scdl/if_dl_status_c=>sc_t_plan_putaway_dist.
ls_status_stm-status_type = ls_status-status_type.           ls_status_stm-
status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
```

```
                IMPORTING
```

Source Coding of Sample Function Module LZHTGP02

```
              es_status_text =  ls_status_text ).
          cs_whrhead-status_plan_putaway_dist
= ls_status_text-status_value_text.

        WHEN /scdl/if_dl_status_c=>sc_t_putaway_dist.
          ls_status_stm-status_type = ls_status-status_type.
ls_status_stm-status_value = ls_status-status_value.
          get_status_text(
            EXPORTING
              is_status_stm = ls_status_stm
IMPORTING                es_status_text =
ls_status_text ).
          cs_whrhead-status_putaway_dist
            = ls_status_text-status_value_text.
ENDCASE.

    ENDLOOP.

  ENDMETHOD.                    "set_status_data
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
* Protected Section
*---------------------------------------------------------------------*
  METHOD add_sel_crit_common.
*     IMPORTING
*     iv_lgnum      TYPE  /scwm/lgnum
*     CHANGING
*     ct_huident_r  TYPE  rseloption
*     ct_selection  TYPE  /scwm/dlv_selection_tab

    DATA:
      ls_asn           LIKE LINE OF  so_asn,
ls_bol           LIKE LINE OF  so_bol,
ls_carrier       LIKE LINE OF  so_carr,
ls_docno         LIKE LINE OF  so_docno,
ls_doctype       LIKE LINE OF  so_docty,
ls_door          LIKE LINE OF  so_door,
ls_erp_id        LIKE LINE OF  so_erpid,
ls_huident_r     TYPE          rsdsselopt,
ls_huno          LIKE LINE OF  so_huno,
ls_manual        LIKE LINE OF  so_manu,
ls_pro           LIKE LINE OF  so_pro,
      ls_select        TYPE          /scwm/dlv_selection_str,
ls_sflo             LIKE LINE OF   so_sflo,         ls_sfprt
LIKE LINE OF   so_sfprt,        ls_stprtf         LIKE LINE OF
so_stprf,         ls_tcdref          LIKE LINE OF   so_tcdrf,
ls_tu            LIKE LINE OF  so_tu.
*     Delivery type
    LOOP AT so_docty INTO ls_doctype.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_doctype TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_doctype_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*     Transport unit
    LOOP AT so_tu INTO ls_tu.
CLEAR ls_select.
```

Source Coding of Sample Function Module LZHTGP02

```
      MOVE-CORRESPONDING ls_tu TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_transmeans_id_h.
      APPEND ls_select TO ct_selection.
ENDLOOP.
*   HU number
    LOOP AT so_huno INTO ls_huno.
CLEAR ls_huident_r.
      MOVE-CORRESPONDING ls_huno TO ls_huident_r.
      APPEND ls_huident_r TO ct_huident_r.
    ENDLOOP.
*   Delivery number
    LOOP AT so_docno INTO ls_docno.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_docno TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_docno_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Manual attribute
    LOOP AT so_manu INTO ls_manual.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_manual TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_manual_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   TCD reference number
    LOOP AT so_tcdrf INTO ls_tcdref.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_tcdref TO ls_select.
      ls_select-fieldname = /scwm/if_dl_logfname_c=>sc_refdocno_tcd_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Bill of lading
    LOOP AT so_bol INTO ls_bol.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_bol TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_refdocno_bol_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Ship from location
    LOOP AT so_sflo INTO ls_sflo.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_sflo TO ls_select.       ls_select-
fieldname =
        /scdl/if_dl_logfname_c=>sc_locationno_sflo_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Ship from party
    LOOP AT so_sfprt INTO ls_sfprt.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_sfprt TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_partyno_sfprt_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Final ship-to party
    LOOP AT so_stprf INTO ls_stprtf.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_stprtf TO ls_select.       ls_select-
fieldname =
```

```
        /scwm/if_dl_logfname_c=>sc_partyno_stprtf_h.
APPEND ls_select TO ct_selection.
```

Source Coding of Sample Function Module LZHTGP02

```
    ENDLOOP.
*   Carrier
    LOOP AT so_carr INTO ls_carrier.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_carrier TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_partyno_carr_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Door
    LOOP AT so_door INTO ls_door.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_door TO ls_select.
      ls_select-fieldname = /scwm/if_dl_logfname_c=>sc_door_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   ASN number
    LOOP AT so_asn INTO ls_asn.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_asn TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_refdocno_asn_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   ASN number
    LOOP AT so_pro INTO ls_pro.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_pro TO ls_select.
      ls_select-fieldname = /scwm/if_dl_logfname_c=>sc_refdocno_pro_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
***********************************************************************
*   Add additional selection criteria for ERP document ID
***********************************************************************
*   ERP document ID
    LOOP AT so_erpid INTO ls_erp_id.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_erp_id TO ls_select.
      ls_select-fieldname = /scdl/if_dl_logfname_c=>sc_refdocno_erp_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
***********************************************************************
  ENDMETHOD.                    "add_sel_crit_common
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD add_sel_crit_status.
*     IMPORTING
*     iv_lgnum       TYPE  /scwm/lgnum
*     CHANGING
*     ct_selection   TYPE  /scwm/dlv_selection_tab.

    DATA:
      lt_logfname_key  TYPE          /scdl/dl_logfname_key_tab,
lt_logfname_map  TYPE          /scdl/dl_logfname_map_tab.
    DATA:       ls_dsiva         LIKE LINE OF
so_dsiva,
```

```
      ls_inparam       TYPE          /scwm/s_sp_qry_logfname_inp,
ls_logfname_key  TYPE          /scdl/dl_logfname_key_str,
ls_logfname_map2 TYPE          /scdl/dl_logfname_map_str,
ls_select     TYPE          /scwm/dlv_selection_str,
```

Source Coding of Sample Function Module LZHTGP02

```
      ls_status_value  LIKE LINE OF  so_dgri.
DATA:
      lv_rejected      TYPE          boole_d.
    DATA:       lo_message_box   TYPE REF TO
/scdl/cl_sp_message_box,       lo_sp_core       TYPE REF TO
/scdl/cl_sp.

*   Status in yard
    LOOP AT so_dtrh INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_dtr_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status goods receipt
    LOOP AT so_dgri INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_dgr_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status putaway
    LOOP AT so_dpti INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_dpt_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status planned putaway
    LOOP AT so_deui INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_deu_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status unloading
    LOOP AT so_duni INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_dun_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status WM activity
    LOOP AT so_dwai INTO ls_status_value.
      CLEAR ls_select.
      MOVE-CORRESPONDING ls_status_value TO ls_select.
      ls_select-fieldname =
        /scdl/if_dl_logfname_c=>sc_status_value_dwa_i.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*   Status NCTS
```

```
LOOP AT so_dwnh INTO ls_status_value.
  CLEAR ls_select.
  MOVE-CORRESPONDING ls_status_value TO ls_select.
  ls_select-fieldname =
```

Source Coding of Sample Function Module LZHTGP02

```
        /scwm/if_dl_logfname_c=>sc_status_value_dwn_h.
      APPEND ls_select TO ct_selection.
    ENDLOOP.
*     Status Blocked Overall     IF NOT p_dboi IS
INITIAL.
      CLEAR ls_select.
      ls_select-fieldname =

/scdl/if_dl_logfname_c=>sc_status_value_dbo_i.
ls_select-sign   = 'I'.       ls_select-option =
'EQ'.       ls_select-low    = p_dboi.       APPEND
ls_select TO ct_selection.
    ENDIF.

*     Dynamic status (item)
    IF po_dsity IS NOT INITIAL.
      ls_inparam-structure = '/SCDL/S_SP_A_ITEM_STATUS'.
ls_inparam-level     = /scdl/if_dl_object_c=>sc_object_level_i.
ls_inparam-fieldname = 'STATUS_VALUE'.       ls_logfname_key-
keyfield = 'STATUS_TYPE'.
      ls_logfname_key-keyvalue = po_dsity.
      APPEND ls_logfname_key TO lt_logfname_key.

      TRY.
          CREATE OBJECT lo_message_box.
        CATCH /scdl/cx_sp_message_box.                  "#EC NO_HANDLER
      ENDTRY.
      CREATE OBJECT lo_sp_core TYPE /scdl/cl_sp_prd_inb
EXPORTING
          io_message_box = lo_message_box           iv_mode
= /scdl/cl_sp_prd_inb=>sc_mode_classic.

*     Query the parameter       lo_sp_core->query(
        EXPORTING inkeys      = lt_logfname_key
inparam     = ls_inparam
                  query       =
/scwm/if_sp_c=>sc_qry_logfname         IMPORTING outrecords
= lt_logfname_map                   rejected    = lv_rejected
).
      IF lv_rejected = abap_true.
        ASSERT ID /scwm/ui_delivery CONDITION 1 = 0.
ENDIF.

*     Prepare selection condition
      READ TABLE lt_logfname_map
        WITH KEY fieldname = 'STATUS_VALUE'
        INTO ls_logfname_map2.
IF sy-subrc NE 0.
        ASSERT ID /scwm/ui_delivery CONDITION 1 = 0.
      ELSE.
        ls_select-fieldname = ls_logfname_map2-logfname.
        LOOP AT so_dsiva INTO ls_dsiva.
          ls_select-sign   = ls_dsiva-sign.
ls_select-option = ls_dsiva-option.
ls_select-low    = ls_dsiva-low.
ls_select-high   = ls_dsiva-high.
APPEND ls_select TO ct_selection.
        ENDLOOP.
```

```
      ENDIF.
ENDIF.
```

Source Coding of Sample Function Module LZHTGP02

```
  ENDMETHOD.                    "add_sel_crit_status
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
  METHOD get_status_text.
*     IMPORTING
*     is_status_stm   TYPE  /scdl/stm_status_str
*     EXPORTING
*     es_status_text  TYPE  /scdl/stm_status_text_str

    CLEAR es_status_text.

    TRY.         go_stm-
>get_short_text(           EXPORTING
            is_status        = is_status_stm
iv_language      = sy-langu           IMPORTING
es_status_text   = es_status_text ).
      CATCH /scdl/cx_stm.                                    "#EC NO_HANDLER
ENDTRY.


  ENDMETHOD.                    "get_status_text
*---------------------------------------------------------------------*
ENDCLASS.                    "lcl_hlp_id IMPLEMENTATION
```

8.1.5 Function Module ZHTG_WHRHEAD_MON_XT

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
FUNCTION zhtg_whrhead_mon_xt.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IV_LGNUM) TYPE  /SCWM/LGNUM
*"     REFERENCE(IV_MODE) TYPE  /SCWM/DE_MON_FM_MODE DEFAULT '1'
*"     REFERENCE(IV_VARIANT) TYPE  VARIANT OPTIONAL
*"  EXPORTING
*"     REFERENCE(ET_DATA) TYPE  ZHTG_TT_WIP_WHRHEAD
*"     REFERENCE(EV_RETURNCODE) TYPE  XFELD
*"     REFERENCE(EV_VARIANT) TYPE  VARIANT
*"  CHANGING
*"     REFERENCE(CT_FIELDCAT) TYPE  LVC_T_FCAT OPTIONAL
*"     REFERENCE(CT_TAB_RANGE) TYPE  RSDS_TRANGE OPTIONAL
*"  RAISING
*"      /SCWM/CX_MON_NOEXEC
*"----------------------------------------------------------------------
-    DATA:
    lt_aggr           TYPE          /scwm/tt_wip_whr_aggr,
lt_aggr2           TYPE           /scwm/tt_wip_whr_aggr2,
lt_docid_tunum   TYPE         lty_t_docid_tu,
    lt_doctype_txt    TYPE           /scwm/dlv_doctype_text_tab,
lt_head            TYPE           /scwm/dlv_header_out_prd_tab,
lt_item          TYPE         /scwm/dlv_item_out_prd_tab,
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
    lt_mapping        TYPE          /scwm/tt_map_selopt2field,
lt_selection      TYPE          /scwm/dlv_selection_tab.
  DATA:
    ls_aggr           TYPE          /scwm/s_wip_whr_aggr,
ls_aggr2            TYPE            /scwm/s_wip_whr_aggr2,
ls_date               TYPE              /scdl/dl_date_str,
ls_docid_tunum    TYPE          lty_docid_tu,
    ls_doctype_txt    TYPE            /scwm/dlv_doctype_text_str,
ls_head             TYPE            /scwm/dlv_header_out_prd_str,
ls_include_data   TYPE            /scwm/dlv_query_incl_str_prd,
ls_item               TYPE            /scwm/dlv_item_out_prd_str,
ls_partyloc           TYPE                /scdl/dl_partyloc_str,
ls_read_options   TYPE                /scwm/dlv_query_contr_str,
ls_refdoc        TYPE        /scdl/dl_refdoc_str,     ls_timedate
TYPE          /scwm/s_tstmp_date_time,      ls_transport      TYPE
/scdl/dl_transport_str,
*    Enhanced structure for EWM Monitor including field for ERP document
ls_whrhead        TYPE          zhtg_s_wip_whrhead.
  DATA:      lv_counter        TYPE
i,
    lv_door              TYPE             /scwm/de_door,
lv_door_set       TYPE          abap_bool,      lv_no_item
TYPE         int4,     lv_no_product    TYPE         int4,
lv_repid          TYPE          sy-repid,      lv_productid
TYPE           /scdl/dl_productid,       lv_unload_point
TYPE        /scdl/dl_locationno,     lv_up_set       TYPE
abap_bool.

DATA:
    lo_prd            TYPE REF TO  /scwm/cl_dlv_management_prd.

  CONSTANTS:      lc_dynnr_0100    TYPE  sydynnr
VALUE '0100',      lc_mode_sel_var TYPE
/scwm/de_mon_fm_mode VALUE '3',      lc_mode_std_sel TYPE
/scwm/de_mon_fm_mode VALUE '1'.

  CLEAR: et_data, ev_returncode, ev_variant.

***********************************************************************
*     STEP 1: Check whether only selection variant of function module
*
*     should be chosen
*
***********************************************************************
  IF iv_mode = lc_mode_sel_var.
CALL FUNCTION 'RS_VARIANT_CATALOG'
EXPORTING
        report                = sy-repid
*     NEW_TITLE             = ' '
        dynnr                 = lc_dynnr_0100
*     POP_UP                = ' '        IMPORTING
        sel_variant           = ev_variant
EXCEPTIONS
        no_report             = 1
report_not_existent  = 2
report_not_supplied  = 3
no_variants          = 4
no_variant_selected  = 5
variant_not_existent = 6
OTHERS               = 7.
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
        WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
    ENDIF.
    RETURN.
  ENDIF.
*********************************************************************
*     Create helper class instance   IF go_hlp_id IS INITIAL.     CREATE
OBJECT go_hlp_id.
  ENDIF.
*********************************************************************
*     STEP 2: Clear dynpro screen elements
*
*********************************************************************
lv_repid = sy-repid.

  CALL FUNCTION
'/SCWM/DYNPRO_ELEMENTS_CLEAR'     EXPORTING
iv_repid = lv_repid.
*********************************************************************

*********************************************************************
*     STEP 3: Map select-options and parameters to database tables and
*
*     fields
*
*********************************************************************
go_hlp_id->map_whrhead_to_sel_opt(
    CHANGING ct_mapping = lt_mapping ).
   p_lgnum =
iv_lgnum.

*********************************************************************

*********************************************************************
*     STEP 4: If variant is provided, fill selection screen in
accordance  *
*     with variant
*
*********************************************************************
  IF NOT iv_variant IS INITIAL.
*     Use the selection criteria from a pre-defined variant without *
presenting a selection screen     CALL FUNCTION 'RS_SUPPORT_SELECTIONS'
EXPORTING
        report                 = sy-repid
variant                     =  iv_variant
EXCEPTIONS
        variant_not_existent = 1
variant_obsolete     = 2
OTHERS               = 3.
    IF sy-subrc <> 0.
      MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.     ENDIF.
  ENDIF.
*********************************************************************

*********************************************************************
* STEP 5: In case selection criteria of previous node are passed fill  *
```

```
* dynpro with them                                                    *
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
*********************************************************************
  IF ct_tab_range IS NOT INITIAL.
*     the table it_tab_range contains the selection criteria, which
*     have been passed to the function module
*     these selection criteria should be visible in the selection screen
CALL FUNCTION '/SCWM/RANGETAB2SELOPT'       EXPORTING
        iv_repid     = sy-repid
iv_lgnum     = iv_lgnum
it_mapping   = lt_mapping
CHANGING         ct_tab_range =
ct_tab_range.
  ENDIF.
*********************************************************************

*********************************************************************
*     STEP 6: If drill-down is used and parent data is passed move keys of *
*     parent data to selection screen -> not used in this function module  *
*********************************************************************
*     CALL FUNCTION '/SCWM/FILL_SELOPT_BY_KEYS'
*     EXPORTING
*     iv_repid       = lv_repid
*     iv_lgnum       = iv_lgnum
*     it_mapping     = lt_mapping *      it_data_parent = it_data_parent.

*********************************************************************

*********************************************************************
*     STEP 7: Check whether selection screen should be displayed            *
*********************************************************************
  IF iv_mode = lc_mode_std_sel.
*     Show selection screen and use the selection criteria entered on
*     the screen. This screen can also be used for definition of a
*     variant (standard functionality of selection-screens)
    CALL SELECTION-SCREEN lc_dynnr_0100 STARTING AT 10 10
ENDING   AT 130 30.
    IF sy-subrc IS NOT INITIAL.
ev_returncode = abap_true.
      RETURN.
    ENDIF.
  ENDIF.
*********************************************************************

*********************************************************************
*     STEP 8: Optionally offer free dynamic selections -> not used in this *
*     function module                                                       *
*********************************************************************
*     CALL FUNCTION '/SCWM/GET_FREE_SELECTIONS'
*     EXPORTING
*     it_tabname     = lt_tabname
*     IMPORTING
*     ev_cancel      = ev_cancel *      et_whereclause = lt_where_clause.
*********************************************************************
*
*     Convert parameter to time stamps   go_hlp_id->convert_date_time_to_ts(
EXPORTING
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
      iv_date_from = p_tudfr
iv_date_to    = p_tudto
iv_lgnum      = iv_lgnum
iv_time_from = p_tutfr
iv_time_to    = p_tutto
IMPORTING       et_timestamp =
so_tudt[] ).

  go_hlp_id->convert_date_time_to_ts(
EXPORTING
      iv_date_from = p_dlvdfr
iv_date_to    = p_dlvdto       iv_lgnum
= iv_lgnum       iv_time_from =
p_dlvtfr       iv_time_to    =
p_dlvtto
    IMPORTING et_timestamp = so_dlvdt[] ).

*********************************************************************
*     STEP 9: Export selection criteria                             *
*********************************************************************
CALL FUNCTION '/SCWM/SELOPT2RANGETAB'     EXPORTING
      iv_repid      = sy-repid
it_mapping    = lt_mapping
IMPORTING       et_tab_range =
ct_tab_range.
*********************************************************************
*     Define which data should be included in query
  go_hlp_id->fill_include_data(
CHANGING       cs_incl_data =
ls_include_data ).

*     Fill selection table ------------------------------------------------
go_hlp_id->fill_selection_table(     EXPORTING
      iv_lgnum = iv_lgnum
CHANGING       ct_selection =
lt_selection ).
*     Adjust selection table for wildcard searches -------------------------
  /scwm/cl_dlv_ui_services=>modify_wildcard_selections(
CHANGING        ct_selections_prd = lt_selection ).
*     Restrict document flow to 1 level ------------------------------------
ls_read_options-docflow_succ_1level_only = abap_true.   ls_read_options-
data_retrival_only       = abap_true.

*     Partner texts   ls_read_options-mdctrl-text = abap_true.

*********************************************************************
*     STEP 10: Execute selection                                    *
*********************************************************************
TRY.
      lo_prd = /scwm/cl_dlv_management_prd=>get_instance( ).
      lo_prd->query(
EXPORTING
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
          it_selection    = lt_selection
          iv_doccat       =
/scdl/if_dl_doc_c=>sc_doccat_inb_prd           is_read_options
= ls_read_options           is_include_data = ls_include_data
IMPORTING
          et_headers       = lt_head
et_items         = lt_item
et_doctype_texts = lt_doctype_txt ).
    CATCH /scdl/cx_delivery.                              "#EC NO_HANDLER
ENDTRY.
*********************************************************************

*     Check result ----------------------------------------------------
----
  IF lt_head IS INITIAL.
    REFRESH et_data.
    RETURN.
  ENDIF.
*********************************************************************
*     STEP 11: Refine selection data
*
*********************************************************************

*     Get status management instance ----------------------------------
----
  IF go_stm IS INITIAL.
    TRY.
        go_stm = /scdl/cl_stm=>get_instance( ).
      CATCH /scdl/cx_stm.                                 "#EC NO_HANDLER
    ENDTRY.
  ENDIF.
*     Get aggregation data --------------------------------------------
----
  REFRESH so_docid.
REFRESH so_doci2.
  so_docid-sign    = 'I'.
so_docid-option  = 'EQ'.
so_doci2-sign   = 'I'.   so_doci2-
option = 'EQ'.   LOOP AT lt_head
INTO ls_head.     lv_counter =
lv_counter + 1.     so_docid-low  =
ls_head-docid.     APPEND so_docid
TO so_docid.     so_doci2-low =
ls_head-docid.     APPEND so_doci2
TO so_doci2.     IF lv_counter =
gc_package_size.
*     Package size for SELECTS reached, start SELECT
      go_hlp_id->select_huref(
         CHANGING ct_aggr = lt_aggr ).
      go_hlp_id->select_tu(
    CHANGING ct_docid_tunum = lt_docid_tunum ).
      REFRESH so_doci2.
lv_counter = 0.
    ENDIF.
  ENDLOOP.

*     Select data for remaining DOCIDs
```

```
  IF NOT so_doci2 IS
INITIAL.     go_hlp_id-
>select_huref(
CHANGING         ct_aggr =
lt_aggr ).
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
    go_hlp_id->select_tu(       CHANGING
ct_docid_tunum = lt_docid_tunum ).
    REFRESH so_doci2.
  ENDIF.

*     Items and products
  CLEAR: ls_timedate, lv_no_item, lv_no_product.
SORT lt_item BY docid product-productid.

  LOOP AT lt_item INTO ls_item.

    IF sy-tabix = 1.
*     Execute only in first loop       ls_aggr2-docid = ls_item-docid.
    ENDIF.

*     Set aggregation data
    IF ls_item-docid NE ls_aggr2-docid.
      ls_aggr2-/scwm/door            = lv_door.
      ls_aggr2-/scwm/unloading_point = lv_unload_point.
      ls_aggr2-no_item               = lv_no_item.       ls_aggr2-
no_product            = lv_no_product.       ls_aggr2-gr_date_plan
= ls_timedate-date.       ls_aggr2-gr_time_plan          =
ls_timedate-time.
      APPEND ls_aggr2 TO lt_aggr2.
ls_aggr2-docid = ls_item-docid.
lv_no_item    = 1.
      CLEAR:
        ls_timedate,
lv_no_product,
lv_door,
lv_door_set,
lv_up_set,
lv_unload_point,
lv_productid.
    ELSE.
*     Item counter       lv_no_item = lv_no_item + 1.
    ENDIF.

*     Set the planned GR date (only once per header)
    IF ls_timedate-date IS INITIAL.
*     Planned delivery date
      READ TABLE ls_item-dates INTO ls_date
        WITH KEY tsttype = /scdl/if_dl_c=>sc_tsttype_sgoodsreceipt
tst_category = /scdl/if_dl_c=>sc_tstcat_plan.
      IF sy-subrc EQ 0.
        go_hlp_id->convert_timestamp(
EXPORTING
            iv_tstfr      = ls_date-tstfr
iv_lgnum      = iv_lgnum
IMPORTING              es_time_date =
ls_timedate ).
      ENDIF.
    ENDIF.

*   Door
    IF lv_door NE '***' AND ls_item-sapext-/scwm/door IS NOT INITIAL.
IF lv_door_set = abap_false.
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
        lv_door      = ls_item-sapext-/scwm/door.
lv_door_set  = abap_true.
      ENDIF.
      IF lv_door NE ls_item-sapext-/scwm/door.
lv_door = '***'.
      ENDIF.
    ENDIF.

*   Unloading point
    IF lv_unload_point NE '***'.
      READ TABLE ls_item-partyloc INTO ls_partyloc
        WITH KEY party_role = /scdl/if_dl_partyloc_c=>sc_party_role_up.
      IF sy-subrc NE 0.
        CLEAR ls_partyloc-party_role.
      ENDIF.
      IF lv_up_set = abap_false.
        lv_unload_point = ls_partyloc-locationno.
lv_up_set       = abap_true.
      ENDIF.
      IF lv_unload_point NE ls_partyloc-locationno.
        lv_unload_point = '***'.
      ENDIF.
    ENDIF.

*     Product counter
    IF NOT ls_item-product-productid IS INITIAL.
IF ls_item-product-productid NE lv_productid.
        lv_no_product = lv_no_product + 1.
lv_productid = ls_item-product-productid.
      ENDIF.
    ENDIF.

  ENDLOOP.
*     Set aggregation data for last loop ----------------------------------
ls_aggr2-/scwm/door            = lv_door.
  ls_aggr2-/scwm/unloading_point = lv_unload_point.   ls_aggr2-
gr_date_plan          = ls_timedate-date.   ls_aggr2-gr_time_plan
= ls_timedate-time.
  ls_aggr2-no_item               = lv_no_item.
ls_aggr2-no_product            = lv_no_product.
APPEND ls_aggr2 TO lt_aggr2.


*     Fill export table ---------------------------------------------------
  LOOP AT lt_head INTO ls_head.
    CLEAR ls_whrhead.

*     Set head data     ls_whrhead-docid   = ls_head-docid.     ls_whrhead-
docno_h = ls_head-docno.     ls_whrhead-doctype = ls_head-doctype.
ls_whrhead-manual  = ls_head-manual.
    MOVE-CORRESPONDING ls_head-eew TO ls_whrhead.           "#EC ENHOK
READ TABLE lt_doctype_txt INTO ls_doctype_txt
      WITH KEY doctype = ls_head-doctype.
    ls_whrhead-doctype_txt   = ls_doctype_txt-text.

    READ TABLE ls_head-transport INDEX 1
      INTO ls_transport.
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
    IF sy-subrc = 0.
      ls_whrhead-transmeans_id = ls_transport-transmeans_id.
ls_whrhead-transpl_type       =    ls_transport-transpl_type.
ENDIF.

*     Set SAPEXT data     ls_whrhead-/scwm/priop = ls_head-sapext-
/scwm/priop.

*     Set reference document data     LOOP AT ls_head-refdoc INTO
ls_refdoc.       CASE ls_refdoc-refdoccat.
*     Bill of lading
        WHEN /scdl/if_dl_c=>sc_doccat_bol.
          ls_whrhead-bol_number = ls_refdoc-refdocno.
*     ASN number
        WHEN /scdl/if_dl_c=>sc_doccat_asn.
          ls_whrhead-refdocno_extasn = ls_refdoc-refdocno.
*     PRO number
        WHEN /scwm/if_dl_c=>sc_doccat_pro.
          ls_whrhead-pro_number = ls_refdoc-refdocno.
*     TCD reference number
        WHEN /scwm/if_dl_c=>sc_doccat_tcd.
          ls_whrhead-tcd_ref_no = ls_refdoc-refdocno.
      ENDCASE.
    ENDLOOP.

*     Set status data
    go_hlp_id->set_status_data(
EXPORTING
        it_status = ls_head-
status       CHANGING
cs_whrhead = ls_whrhead ).

*     Set partyloc data
    LOOP AT ls_head-partyloc INTO ls_partyloc.
      CASE ls_partyloc-party_role.
*     Receiving office
        WHEN /scdl/if_dl_c=>sc_party_role_ro.
          ls_whrhead-receiving_office = ls_partyloc-orgunitno.
*     Carrier
        WHEN /scdl/if_dl_c=>sc_party_role_carr.
          ls_whrhead-carrier = ls_partyloc-partyno.           ls_whrhead-
carrier_descr = ls_partyloc-party_text.
*     Ship from party
        WHEN /scdl/if_dl_c=>sc_party_role_sfprt.
          ls_whrhead-ship_from_party = ls_partyloc-partyno.
ls_whrhead-ship_from_party_descr = ls_partyloc-party_text.
*     Final ship to party
        WHEN /scwm/if_dl_c=>sc_party_role_stprtf.
          ls_whrhead-final_ship_to_party = ls_partyloc-partyno.
          ls_whrhead-final_ship_to_party_descr = ls_partyloc-party_text.
*     Ship from location
        WHEN /scdl/if_dl_c=>sc_party_role_sflo.
          ls_whrhead-ship_from_location = ls_partyloc-locationno.
          ls_whrhead-ship_from_location_descr = ls_partyloc-location_text.
      ENDCASE.
    ENDLOOP.
```

```
*      Set TU data
    LOOP AT lt_docid_tunum INTO ls_docid_tunum
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
      WHERE docid = ls_head-docid.
IF ls_whrhead-tu IS INITIAL.
        ls_whrhead-tu = ls_docid_tunum-tu_num_ext.
      ELSE.
        ls_whrhead-tu = '***'.
        EXIT.
      ENDIF.
    ENDLOOP.

*   Set aggregation data
    READ TABLE lt_aggr INTO ls_aggr
WITH KEY docid = ls_head-docid.
    IF sy-subrc EQ 0.
      ls_whrhead-no_hu = ls_aggr-no_hu.
    ENDIF.
    READ TABLE lt_aggr2 INTO ls_aggr2 WITH KEY docid = ls_head-docid.
    IF sy-subrc EQ 0.
      ls_whrhead-/scwm/door            = ls_aggr2-/scwm/door.
ls_whrhead-/scwm/unloading_point = ls_aggr2-/scwm/unloading_point.
ls_whrhead-no_item               = ls_aggr2-no_item.       ls_whrhead-
no_product            = ls_aggr2-no_product.       ls_whrhead-
gr_date_plan          = ls_aggr2-gr_date_plan.       ls_whrhead-
gr_time_plan          = ls_aggr2-gr_time_plan.     ENDIF.

*   Planned delivery date
    READ TABLE ls_head-dates INTO ls_date
      WITH KEY tsttype = /scdl/if_dl_c=>sc_tsttype_delivery
tst_category = /scdl/if_dl_c=>sc_tstcat_plan.     IF sy-subrc
EQ 0.

      go_hlp_id->convert_timestamp(
EXPORTING
           iv_tstfr      = ls_date-tstfr
iv_lgnum      = iv_lgnum
IMPORTING             es_time_date =
ls_timedate ).
       ls_whrhead-dlv_date = ls_timedate-
date.       ls_whrhead-dlv_time =
ls_timedate-time.     ENDIF.

*   Ship date
    READ TABLE ls_head-dates INTO ls_date
      WITH KEY tsttype = /scdl/if_dl_c=>sc_tsttype_outyard
tst_category = /scdl/if_dl_c=>sc_tstcat_plan.
    IF sy-subrc EQ 0.
      go_hlp_id->convert_timestamp(
EXPORTING
           iv_tstfr      = ls_date-tstfr
iv_lgnum      = iv_lgnum
IMPORTING             es_time_date =
ls_timedate ).
       ls_whrhead-ship_date = ls_timedate-
date.       ls_whrhead-ship_time =
ls_timedate-time.     ENDIF.
***********************************************************************
*   Add information about ERP document to result list set
```

Source Coding of Sample Function Module ZHTG_WHRHEAD_MON_XT

```
*********************************************************************
    CLEAR: ls_refdoc.
    READ TABLE ls_head-refdoc INTO ls_refdoc
      WITH KEY refdoccat = /scwm/if_doc_c=>sc_doccat_erp.
    IF sy-subrc = 0.
      ls_whrhead-erp_docno = ls_refdoc-refdocno.
    ENDIF.
*********************************************************************
    APPEND ls_whrhead TO et_data.
  ENDLOOP.

  SORT et_data BY docno_h.

* Change order of columns in ALV result list
  go_hlp_id->modify_field_cat(
CHANGING       ct_fieldcat =
ct_fieldcat ).

ENDFUNCTION.
```

8.1.6 Text Symbols and Selection Texts of Function Group ZHTG

Maintain text symbols and selection texts for function group ZHTG: Start transaction SE38 and enter the master program SAPLZHTG. Choose option *Text elements* and maintain the following entries:

Tabstrip *Text Symbol*:

Sym	Text	dLen	mLen
001	Header Data	11	20
003	To	2	30
006	No SC unit found for the warehouse number	41	50
007	Selection Period TU	19	25
020	Status (Item)	13	20
030	Data Source	11	11

Tabstrip *Selection Text*:

Name	Typ	Dictionary ref.
PO_DSITY	Status Type	X
P_ARCH	Archive	X
P_BOTH	Database and Archive	X
P_DB	Database	X
P_DBOI	Locked (Overall, Item)	
P_DLVDFR	Planned Delivery Date (From)	

P_DLVDTO	Planned Delivery Date (To)	
Name	**Typ**	**Dictionary ref.**
P_DLVTFR	Planned Delivery Time (From)	
P_DLVTTO	Planned Delivery Time (To)	
P_TUDFR	Selection Date TU (From)	
P_TUDTO	Selection Date TU (To)	
P_TUTFR	Selection Time TU (From)	
P_TUTTO	Selection Time TU (To)	
SO_ASN	ASN	X
SO_BOL	Bill of Lading	X
SO_CARR	Carrier	X
SO_DEUI	Putaway Stat. (Plan, Item)	
SO_DGRI	Goods Receipt Status (Item)	
SO_DOCNO	Inbound Delivery	X
SO_DOCTY	Document Type	X
SO_DOOR	Warehouse Door	X
SO_DPTI	Putaway Status (Item)	
SO_DSIVA	Status Value	X
SO_DTRH	Transit Status	X
SO_DUNI	Unload (Item)	
SO_DWAI	Whse Activity Status (Item)	
SO_DWNH	Transit Procedure	X
SO_ERPID	ERP Document	X
SO_HUNO	Handling Unit	X
SO_MANU	Generated Manually	X
SO_PRO	PRO Number	X
SO_SFLO	Ship-From Location	X
SO_SFPRT	Ship-from	X
SO_STPRF	Final Ship-To Party	X
SO_TCDRF	TCD Process Number	X
SO_TU	Vehicle	X

SO_TU_EX	Transportation Unit	X

8.1.7 Function Module ZHTG_WAVEMON_UNASSIGN _XT– Extended Check for Typing of IT_DATA

This variant of the function module contains an additional check for the type of the importing parameter IT_DATA. IT_DATA uses a generic type (STANDARD TABLE). Using generic types may cause problems if the function module importing parameter gets values of a different type. Checking the types of internal tables can be executed using the class CL_ABAP_TYPEDESCR. The following source code contains an implemented check for proper typing of the importing parameter values:

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN_XT

```
FUNCTION zhtg_wavemon_unassign_xt.
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(IV_LGNUM) TYPE  /SCWM/LGNUM
*"     REFERENCE(IT_DATA) TYPE  STANDARD TABLE
*"----------------------------------------------------------------------
-    DATA:     lt_bapiret           TYPE bapiret2_t,
    lt_odo_item_xt                   TYPE   zhtg_tt_wip_whritem_out,
lt_wave_unass        TYPE /scwm/tt_wave_itm.
  DATA:
    ls_display_profile   TYPE bal_s_prof,     ls_log
TYPE bal_s_log,     ls_odo_item_xt       TYPE
zhtg_s_wip_whritem_out.
  DATA:     lv_lines_w_wave      TYPE i,
lv_loghandle         TYPE balloghndl,
lv_selected_lines    TYPE i,
lv_tab_name          TYPE string.
  DATA:     lo_log               TYPE REF TO
/scwm/cl_log,     lo_type_descr        TYPE REF TO
cl_abap_typedescr.

  CONSTANTS:     lc_doccat_pdo        TYPE /scwm/de_doccat VALUE
'PDO',     lc_ttype_odo_item_xt TYPE string VALUE
'ZHTG_TT_WIP_WHRITEM_OUT'.
  FIELD-SYMBOLS:
    <ls_data> TYPE any.

  CREATE OBJECT lo_log.

********************************************************************** *
Check that entries have been selected from result list
**********************************************************************
lv_selected_lines = lines( it_data ).
  IF lv_selected_lines < 1.
*     No data transferred -> stop execution
    MESSAGE s050(/scwm/wave) DISPLAY LIKE 'E'. *
No delivery items were transferred
    RETURN.
  ENDIF.
**********************************************************************
*     Move data into table which can be used for further processing
*
**********************************************************************
*     Check type of IT_DATA   lo_type_descr =
cl_abap_structdescr=>describe_by_data( it_data ).
  lv_tab_name = lo_type_descr->get_relative_name( ).

  IF lv_tab_name = lc_ttype_odo_item_xt.
*     Type of IT_DATA is equal to type used for result list: Direct
*     transfer of complete table is possible     lt_odo_item_xt[] =
it_data[].
  ELSE.
*     Type of IT_DATA is different from expected table type: Transfer *
necessary table data
```

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN_XT

```
    LOOP AT it_data ASSIGNING <ls_data>.
      MOVE-CORRESPONDING <ls_data> TO ls_odo_item_xt.
      APPEND ls_odo_item_xt TO lt_odo_item_xt.
    ENDLOOP.
  ENDIF.
* Determine wave items of waves in status 'initial' (I) or 'hold' (H)
* that belong to selected outbound delivery order items
  SELECT m~lgnum m~wave m~wave_itm
    FROM /scwm/waveitm AS m
      INNER JOIN /scwm/wavehdr AS t ON
        ( t~wave = m~wave AND t~lgnum = m~lgnum )
      INTO CORRESPONDING FIELDS OF TABLE lt_wave_unass
    FOR ALL ENTRIES IN lt_odo_item_xt
      WHERE m~lgnum = iv_lgnum AND
m~docno = lt_odo_item_xt-docno_h AND
m~itemno = lt_odo_item_xt-itemno AND
( t~status = wmegc_stwave_initial OR
t~status = wmegc_stwave_hold ).
   lv_lines_w_wave = lines( lt_wave_unass
).

  IF lv_lines_w_wave = 0.
*     No data that contains wave -> stop processing
    MESSAGE s008(/scwm/wave) DISPLAY LIKE 'E'. *
No items found for removal
    RETURN.
  ELSE.
    IF lv_lines_w_wave < lv_selected_lines.                 "#EC NEEDED *
Info: Not all selected entries were assigned to waves and could
*     be unassigned -> add custom info message
    ENDIF.
  ENDIF.
**********************************************************************
*     Process data                                                   *
**********************************************************************
CALL FUNCTION '/SCWM/WAVE_ITEMS_UNASSIGN_EXT'     EXPORTING
      iv_lgnum       = iv_lgnum
iv_rdoccat     = lc_doccat_pdo
it_wave_unass  = lt_wave_unass
iv_update_task = 'X'
iv_commit_work = 'X'     IMPORTING
et_bapiret     = lt_bapiret.

  IF NOT lt_bapiret[] IS INITIAL.
*     Handle messages that occurred during function processing     lo_log-
>add_log( it_prot = lt_bapiret ).
    ls_log-extnumber = 1.     ls_log-object
= wmegc_apl_object_wme.     ls_log-
subobject = wmegc_apl_subob_gen.
     lo_log-
>create_log(
EXPORTING
        is_log       = ls_log
IMPORTING         ev_loghandle =
lv_loghandle ).
```

Source Coding of Sample Function Module ZHTG_WAVEMON_UNASSIGN_XT

```
    lo_log->convert_bapiret2applog( ).

*   Get profile for popup application log
    CALL FUNCTION 'BAL_DSP_PROFILE_POPUP_GET'
IMPORTING         e_s_display_profile =
ls_display_profile.
     ls_display_profile-use_grid =
'X'.

    TRY.         lo_log-
>display_log(
EXPORTING
            iv_loghandle       = lv_loghandle
is_display_profile = ls_display_profile ).

      CATCH /scwm/cx_basics.                            "#EC NO_HANDLER
    ENDTRY.
  ENDIF.

ENDFUNCTION.
```